MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVA

DARIO HORVAT

Wi-Fi UPRAVLJANJE KUĆNIM UREĐAJIMA POMOĆU SMARTPHONEA

ZAVRŠNI RAD

Mentor:

Jurica Trstenjak, dipl. ing.

ČAKOVEC, 2014.

SADRŽAJ

SAŽETAK..... 2
1. UVOD..... 3
2. POČETNA IDEJA RAZRADE PROJEKTA..... 4
3. OPIS OSNOVNIH ELEMENATA ZA PROJEKAT..... 5
3.1. Pametni telefon baziran na android platformi..... 5
3.2. Usmjerivač..... 6
3.3. Spark Core..... 7
3.4. Mikrokontroler..... 9
3.5. Wi-Fi Modul CC3000..... 9
3.6. Spark Cloud..... 10
4. IZRADA PROJEKTA..... 11
4.1. Povezivanje jezgre sa usmjerivačem i internetom..... 11
4.2. Programiranje mikrokontrolera..... 15
4.3. Testiranje ispravnosti ulazno-izlaznih signala serijskom usb vezom..... 17
4.4. Izrada probnih shema sklopa za upravljanje strujnim krugom..... 19
4.5. Testiranje probnih shema na eksperimentalnoj pločici..... 21
4.6. Problem sheme 2 i pronalazak rješenja..... 24
4.7. Izrada glavne sheme cijelog sklopa..... 27
4.8. Izrada glavnog sklopa..... 28
4.9. Cjeloviti programski kod za mikrokontroler..... 30
4.10. Izrada probne web aplikacije..... 32
4.11. Izrada mobilne aplikacije za android platformu..... 40
5. TESTIRANJE MOBILNE APLIKACIJE I GOTOVOG HARDVERA..... 41
6. PREZENTACIJSKI PROTOTIP UPRAVLJANJA RASVJETOM..... 44
7. ZAKLJUČAK..... 45
8. LITERATURA..... 46

SAŽETAK

Kao što i sam naziv projekta govori upravljanje kućnim uređajima putem bežične veze (engl. Wi-Fi) pomoću pametnog telefona, tako se i doslovno misli da će se sa pametnim mobilnim uređajem na kojem se nalazi aplikacija za upravljanje kućnim uređajima upravljati sa istima, gdje će se cijeli pozadinski proces komunikacije odvijati putem interneta. Potrebni uređaji i elementi za realizaciju ovoga projekta su pametni telefon baziran na android platformi sa aplikacijom za upravljanje kućnim uređajima, koja će se izraditi u programu poput Eclipse, za izradu mobilnih aplikacija. Također potreban je usmjerivač sa pristupom na internet i bežičnom komunikacijom, elektronički sklop (engl. Spark Core) koji sadrži mikrokontroler i Wi-Fi modul sa dodatnim sklopovima za napajanje, upravljanje određenim trošilom i za integraciju u strujni krug, te program za pisanje koda mikrokontrolera.

1. UVOD

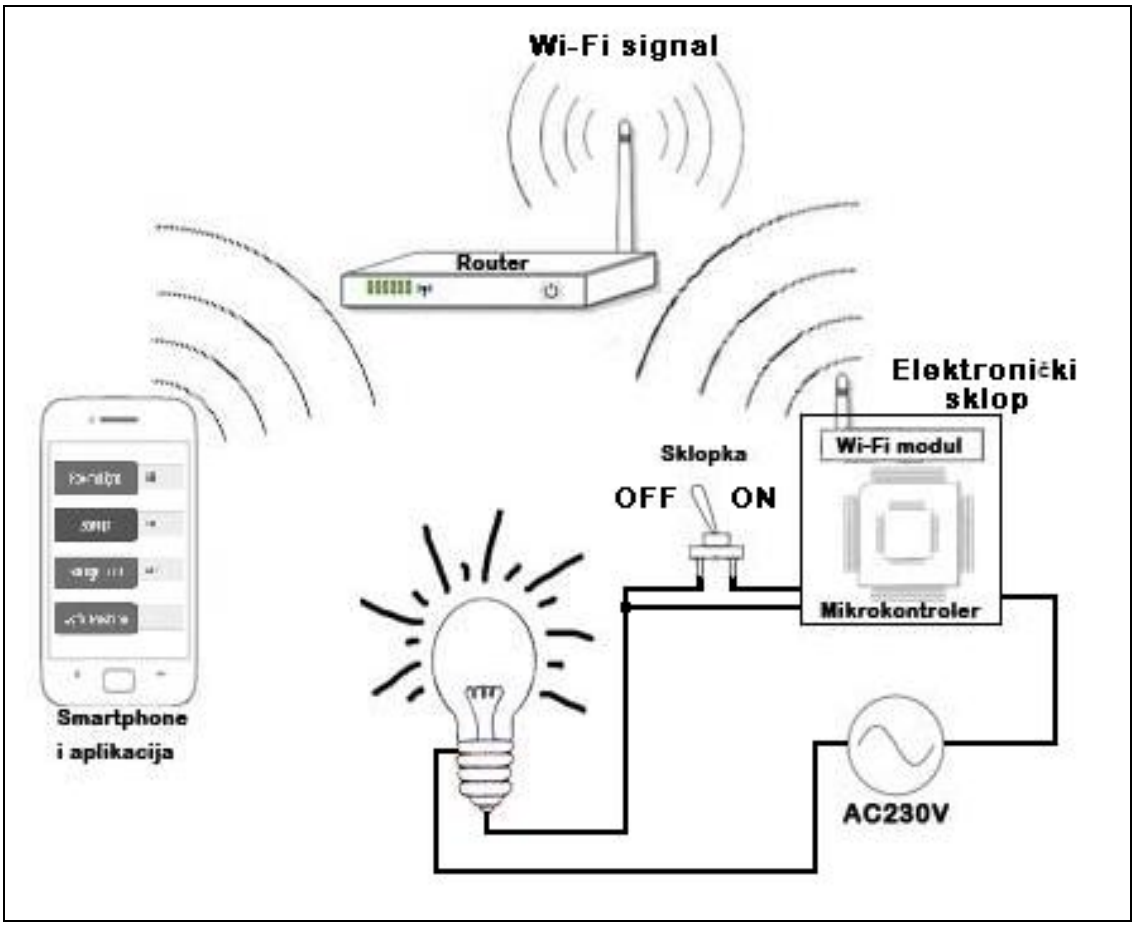
U današnje vrijeme teži se prema razvoju pametnih uređaja i njihovim upravljanjem sa jednog mjesta, točnije upravljanje sa mobilnim uređajima poput pametnih telefona (engl. *Smartphone*), tablet računala ili drugih.

Ujedno upravljanje kućnim uređajima putem bežične veze (engl. *Wi-Fi*) pomoću smartphonea pojednostavljuje svakodnevni život čovjeka, kao što je na primjer jutarnje buđenje gdje se većinom koriste pametni telefoni umjesto uobičajenih budilica, te su oni jedna od prvih stvari koje čovjek današnjice uzima u ruke, čak i prije samog ustajanja iz kreveta. Samim time što većina ljudi nakon ustajanja ujutro uključi svijetla u potrebnim prostorijama, također uključuju aparat za kuhanje jutarnje kave i već slične potrebne uređaje koji se podrazumijevaju kao kućni elektronički uređaji, dakle izradom ovog projekta sve te svakodnevne stvari moći će se upravljati putem pametnog telefona, točnije sa njegovom aplikacijom. Na taj način ne samo da interesantno zvuči uključivanje i isključivanje svijetla, gdje god je ono potrebno bez da se maknemo sa mjesta kao i uključivanje aparata za kavu za vrijeme kad se osoba još nalazi u krevetu, pa tako i upravljanje tv ili radio uređajima te mnoge druge stvari, znači moguće bi bilo još dodatno pojednostavniti svakodnevni život običnog čovjeka. Tako bi mogli zapravo reći da bi pametni telefon postao daljinski upravljač kućnih uređaja.

2. POČETNA IDEJA RAZRADE PROJEKTA

Kako i većina projekata koji započinju nekom vrstom ideje, te prenošenje iste na papir kako bi se vizualno dočarale i pridobile neke smjernice kao, na koji način krenuti sa ostvarivanjem i izradom projekta, tako i slika 1. prikazuje jednostavnu shemu ideje koja predstavlja sve što je potrebno i kako bi funkcioniralo.

Slika 1. *Shema projektne ideje*



3. OPIS OSNOVNIH ELEMENATA ZA PROJEKAT

3.1. Pametni telefon baziran na android platformi

Pametni telefon (engl. *Smartphone*) je zapravo uređaj koji kombinira funkcije osobnog digitalnog asistenta (engl. *PDA*) i običnog mobitela. Karakteristični su radi svoje višezadaćne funkcije, sa ekranima na dodir visoke rezolucije, pristupom internetu preko bežične veze (engl. *Wi-Fi*) ili 3G[I] mreže, multimedijske funkcije kao kamere i reproduciranje (engl. *player*) videozapisa i mp3[II] pjesama, kalendar, upravljanje kontaktima, GPS[III] navigacija, mogućnost čitanja poslovnih dokumenata u različitim formatima kao što su PDF[IV], Microsoft Office i drugi, te ostale raznovrsne funkcionalnosti. Slika 2. prikazuje pametni telefon baziran na android platformi na kojem će se izvršavati aplikacija izrađena u projektu.

Slika 2. *Smartphone Samsung Galaxy SIII mini*

[1]

[I] 3G – Skupno ime za treću generaciju mobilne telefonije, omogućava prijenos podataka do nekoliko desetaka Mbit/s.

[II] mp3 – MP3 ili MPEG-1 Audio Layer 3 je najraširenij lossy audio format.

[III] GPS – Sistem za određivanje pozicije na zemlji (engl. *Global Positioning System*).

[IV] PDF – Format zapisa dvodiomenzionalnih dokumenata (engl. *Portable Document Format*), neovisno od uređaja i rezolucije ispisa.

Kratak popis tehničkih karakteristika pametnog telefona koji će se koristiti:

- Dimenzije: 121.6 x 63 x 9.9 mm,
- Masa: 112 grama,
- Zaslon: 4,0“ Super AMOLED, 480 x 800 točaka, 233 ppi,
- SoC: NovaThor U8420,
- CPU: Dvojezgreni ARM Cortex-A9 1 GHz,
- GPU: Mali400MP,
- RAM: 1 GB,
- OS: Android 4.1 Jelly Bean,
- Interna memorija: 8/16 GB,
- Kamera: 5,0 MP,
- Video kamera: 720p @ 30fps,
- FM Radio s RDS-om,
- Baterija: 1500 mAh.

Detaljniji opis i tehničke karakteristike vezane uz pametni telefon mogu se naći na internetu [2].

3.2. Usmjerivač

Usmjerivač (engl. *Router*) je uređaj koji usmjerava podatkovne pakete na njihovom putu kroz računalnu mrežu pri čemu se taj proces odvija na mrežnom sloju OSI modela[V]. U današnje vrijeme najčešće se govori o usmjerivačima u IP[VI] mrežama, iz razloga što je njihova primjena najraširenija upravo za potrebe najveće računalne mreže današnjice, drugim riječima interneta. Osnovni zadatak koji usmjerivači obavljaju je da za svaki paket koji pristigne na neko od mrežnih sučelja na usmjerivaču provjere odredišnu IP adresu, u svojoj tablici usmjeravanja pronađu kamo treba preusmjeriti taj paket te ga onda proslijede na odgovarajuće sučelje. Na slici 3. prikazan je jednostavan usmjerivač, danas dostupan svakome kućanstvu i koji je upotrijebljen u ovome projektu.

[V] OSI model – Referentni model umrežavanja otvorenih sustava.
[VI] IP (engl. *Internet Protocol*) – Mrežni protokol za prijenos podataka kojeg koriste izvorišna i odredišna računala za uspostavu podatkovne komunikacije preko računalne mreže.

Slika 3. *Usmjerivač sa Wi-Fi pristupom*

3.3. Spark Core

Spark jezgra (engl. *Spark Core*) je zapravo naziv za elektronički sklop prikazan na slici 4. Producirana od strane Spark Devices, a to je tvrtka koja se bavi izradom povezanih uređaja koji su jednostvani, snažni i zabavni. Otvorenog je koda, kompatibilna pločica sa arduinom[VII], dizajnirana tako da uvelike pojednostavljuje posao dodavanja bežične veze s internetom sa vlastitim gizmosima (uređaji ili kontrole koji su vrlo korisni za određeni posao) i patentima, tzv. DIY[VIII] projektima. Riječ Core zapravo dolazi od "Constrained RESTful Environments (CoRE)", drugim riječima razvija okvir za usmjerene resurse aplikacijama namijenjene za rad na ograničene IP mreže, te je podrška ostvarivanju RESTful arhitekture u prikladnom obliku za najviše ograničene čvorove i mreže. Na temelju polja zaglavlja HTTP[IX] veze definiranih u RFC 5988[X], oblik veze "Constrained RESTful Environments" (CoRE) prenosi se kao sadržaj koji je dodijeljen vrsti internetskog medija. RESTful se odnosi na arhitekturu reprezentacijskog prijenosnog stanja (engl. *Representational State Transfer, REST*) [3]. Spark jezgra će zapravo predstavljati mozak cijeloga projekta.

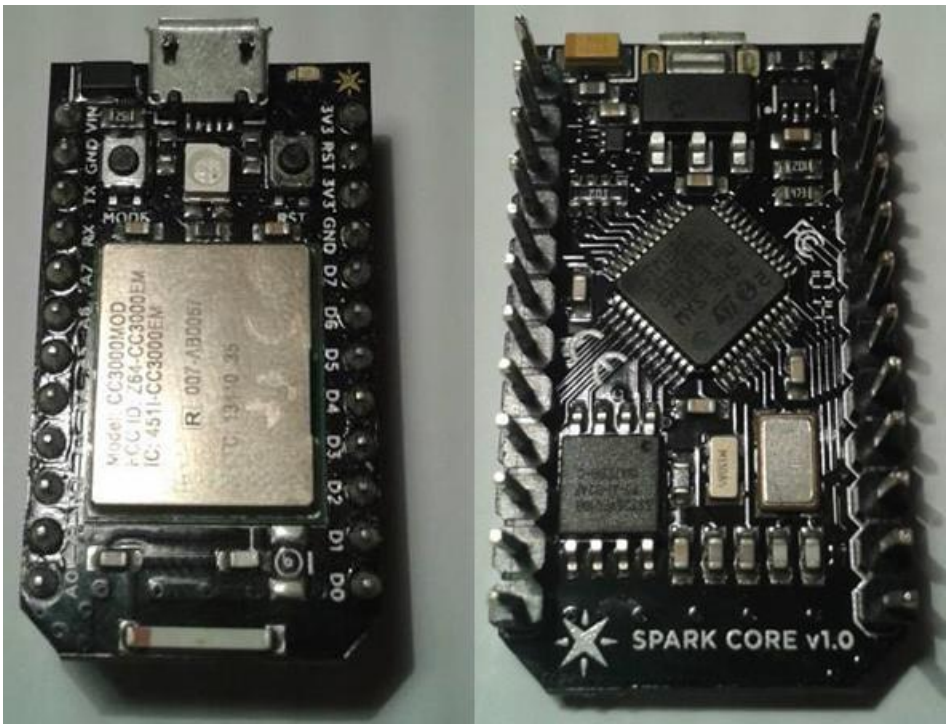
[VII] Arduino – mikrokontroler izgrađen na jednoj pločici, otvorenog koda, svrha mu je da učini primjenu interaktivnih objekata ili okruženja više pristupačnijima. Namijenjen je umjetnicima, dizajnerima, hobistima, te svim zainteresiranima za izradu interaktivnih objekata ili okruženja.

[VIII] DIY – (engl. *Do It Yourself*), uradi sam.

[IX] HTTP – Glavna i najčešća metoda prijenosa informacija na webu (engl. *HyperTextTransferProtocol*).

[X] RFC 5988 – Redefinira vezu zaglavlja, pružajući strukturirani jezik za povezivanje na druge izvore.

Slika 4. *Elektronički sklop Spark Core*



Važne komponente jezgre jesu 72MHz ARM Cortex M3 mikroprocesor, Wi-Fi modul, te usb sučelje. Ulazni i izlazni pinovi mikrokontrolera omogućavaju sučelju upravljanje i komunikaciju sa ciljnim hardverom.

Korisnik razvija program (engl. *Software*) za Spark Core sa popularnim Wiring[XI] programskim okvirom (engl. *Framework* – apstrakcija u kojoj se softversko pružanje generičke funkcionalnosti može selektivno mijenjati dodatnim kodom od strane korisnika, čime se osigurava specifični softver za aplikaciju), koji je jednostavan za naučiti i za upotrebu, te se napisani kod šalje (engl. *Upload*) na njegov ugrađeni program (engl. *Firmware*) putem Wi-Fia ili usb sučelja.

[XI] Wiring – elktronička prototipna platforma otvorenog koda sastavljena od programskog jezika, integriranog razvojnog okruženja (IDE) i mikrokontrolera na jednoj pločici.

3.4. Mikrokontroler

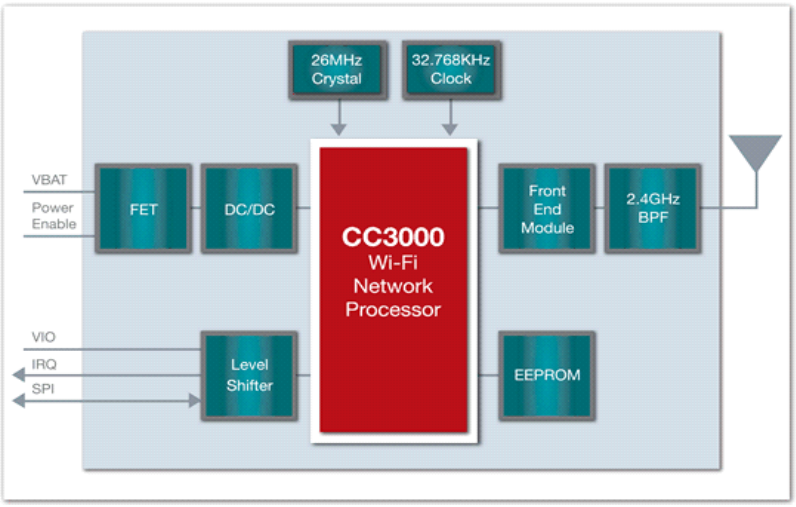
Mikrokontroler koji će se programirati nalazi se kao integracija na sklopu Spark Core. To je STM32F103CB mikrokontroler visokih preformansi baziran na ARM[XII] 32-bitnom cortex M3 procesoru, koji služi kao snažni mozak toga sklopa. Više o mikrokontroleru moguće je proučiti na referenci [4].

3.5. Wi-Fi Modul CC3000

TI[XIII] CC3000 modul je samoodrživi bežični mrežni procesor koji pojednostavljuje implementaciju internetske povezivosti. TI SimpleLink Wi-Fi solucija minimizira softverske zahtjeve domaćina mikrokontrolera MCU[XIV], te je stoga idealno rješenje za ugrađene aplikacije koje koriste mikrokontroler (koji mogu biti jeftini i male potrošnje).

TI CC3000 modul smanjuje vrijeme razvoja i troškove proizvodnje, ne zahtjeva mnogo prostora na elektroničkoj ploči, olakšava certifikaciju, te smanjuje potrebu za RF[XV] stručnošću. Na slici 5. prikazan je dijagram modula CC3000 [5].

Slika 5. *Dijagram modula CC3000*



[6]

[XII] ARM – Ime za vrstu mikroprocesora koji danas pokreću gotovo sve današnje mobitele, dobar dio tableta, kalkulatora, video konzola itd, a koji se temelje na dizajnu britanske korporacija ARM holdings koja ne proizvodi ove procesore nego ih samo dizajnira i potom ubire licencu od proizvođača.
[XIII] TI – Kratica od Texas Instruments, globalno poduzeće dizajna i proizvodnje poluvodiča.
[XIV] MCU – Skraćenica za mikrokontroler, malo računalo na jednom integriranom krugu koji sadrži jezgru procesora, memoriju i programabilne ulazno / izlazne periferije.
[XV] RF – Kratica za radio frekvenciju.

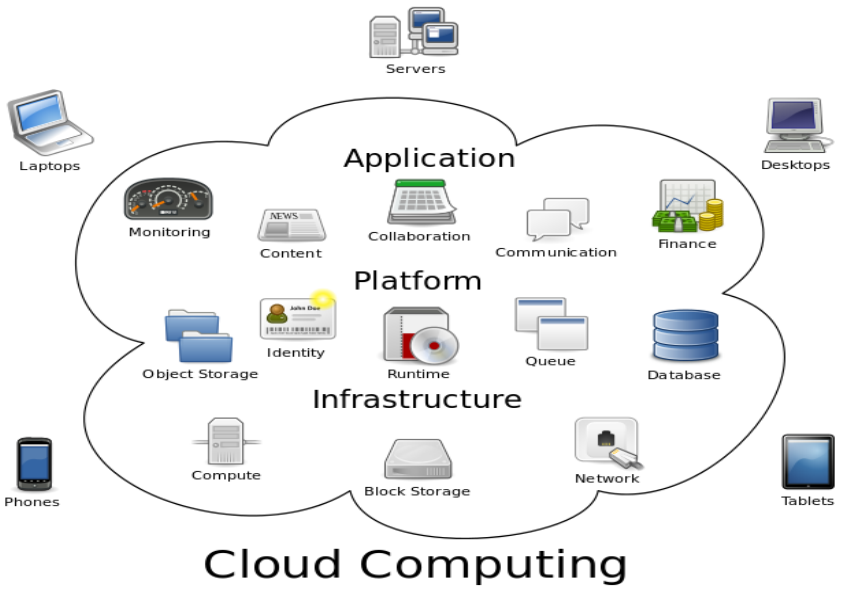
3.6. Spark Cloud

Spark jezgra koristi tzv. uslugu u oblaku (engl. *The Cloud*) što znači da su programi, podaci i aplikacije instalirani na nekom od poslužitelja u podatkovnom centru ponuđača, a to podrazumijeva centralizaciju podataka (sadržaj na jednom mjestu), stalnoj dostupnosti podataka (mobilnost korisnika), manjem riziku od gubitka podataka (politika arhiviranja), boljem korisničkom iskustvu, uvijek zadovoljenom kapacitetu usluga, sigurnosti podataka (bilježi se promjena zapisa, izvoz podataka). Može se reči da se koristi vanjska (engl. *Outsource*) usluga s fiksnim troškovima, a najčešće mjerljivim rezultatima. Za online usluge, programe, sadržaje te njihovo djelovanje angažirani su stručnjaci, koji razvijaju, nadograđuju i održavaju usluge kako bi zadovoljili potrebe korisnika, a kao rezultat toga korisnici usluga u oblaku mogu bolje, jeftinije i jednostavnije komunicirati, izmjenjivati sadržaje i mnoge druge mogućnosti.

Dakle za oblak u ovome projektu može se reći da je kao matični brod na koji se spark core povezuje kada pristupi vezi (engl. *Online*), te dovodi jezgru na stvarni internet. Jezgra uparena sa oblakom postaje dostupna sa bilo kojeg mjesta u svijetu putem otvorenog REST API-a[XVI]. To također stvara sigurno okruženje bez pritiska na web preglednik i jezgru da komuniciraju sa istim programskim jezikom. Slika 6. prikazuje metaforu usluge u oblaku gdje mrežni elementi koje predstavljaju pružatelji datih usluga za korisnika su nevidljivi, kao da su zaklonjeni oblakom [7].

[XVI] REST API – Reprezentacijski prijenos stanja (engl. *Representational State Transfer*) je način za stvaranje, čitanje, ažuriranje i brisanje podataka na poslužitelju koristeći jednostavne pozive. API - Aplikacijsko programsko sučelje.

Slika 6. *Vizualni prikaz usluge u oblaku*



[8]

Bez neke vrste cloud usluge, uređajima povezanih internetom moguće je jedino pristupiti na lokalnoj mreži, osim ukoliko se ne napravi tuneliranje na lokalnoj mreži, što često znači bavljenje vatrozidima i mapiranju portova.

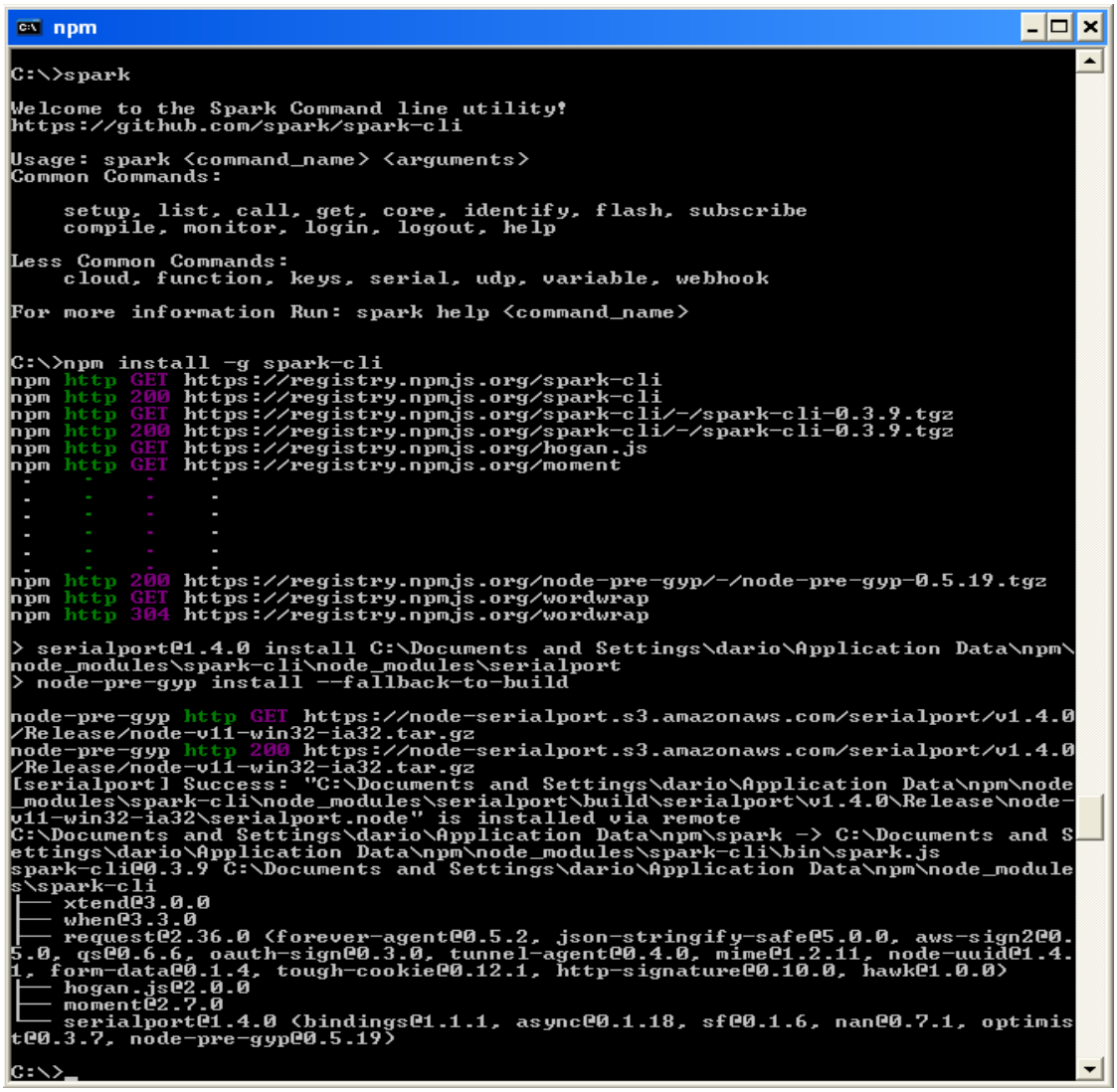
4. IZRADA PROJEKTA

4.1. Povezivanje jezgre sa usmjerivačem i internetom

Uspostava komunikacije putem sučelja naredbenog retka za Spark jezgru (engl. *Spark CLI*), to je zapravo snažan alat za interakciju sa vlastitom jezgrom i spark uslugom u oblaku.

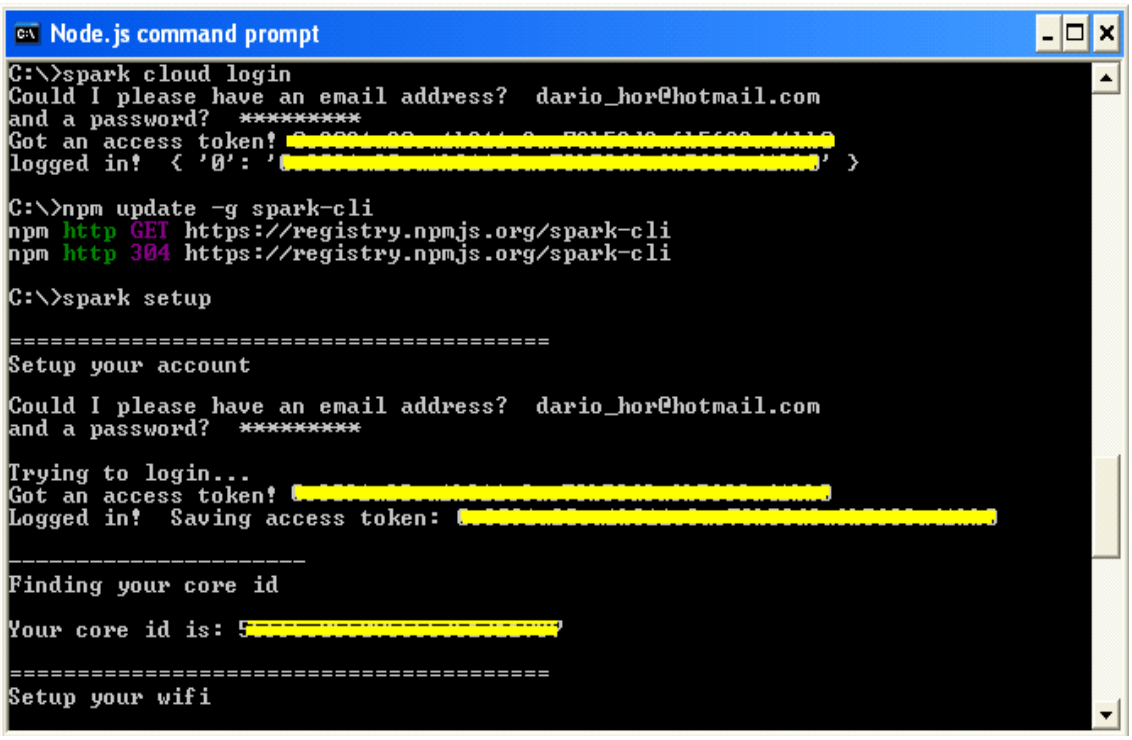
- Potrebno je napraviti instalaciju putem Microsoftovog tumača naredbenog retka baziranog na Windowsovoj platformi (engl. *Windows Comand Prompta*), izvršno ime `cmd.exe` ili putem Node.js softverske platforme za podesivu poslužiteljsku stranu i mrežne aplikacije koja podržava sve operativne sisteme, dakle pozivanjem jednostavne naredbe `$ npm install -g spark-cli` započinje se potrebna instalacija koja odradi cijeli proces za korisnika, što je i prikazano na slici 7.

Slika 7. *Proces instalacije na poziv naredbe*



- Nakon instalacije, naredbom `$ spark cloud login` i unosom traženih podataka vrši se povezivanje sa samim oblakom odakle se dobiva pristupni token koji sadrži sigurnosne vjerodajnice za prijavu i identifikaciju korisnika, korisničku grupu, korisničke privilegije, te u nekim slučajevima i određene aplikacije. Nadalje naredbom `$ spark setup` kreira se korisnički račun odakle se dobiva ID jezgre (ID – funkcija identifikacije je mapiranje poznatog iznosa u nepoznatom subjektu na način da postane poznato. Poznata količina naziva se identifikator ili ID, a nepoznata osoba je ono što treba identifikaciju. Osnovni uvjet za identifikaciju je da bude jedinstveni ID), slika 8. prikazuje logiranje u oblak i postavljanje sparka (engl. *spark setup*).

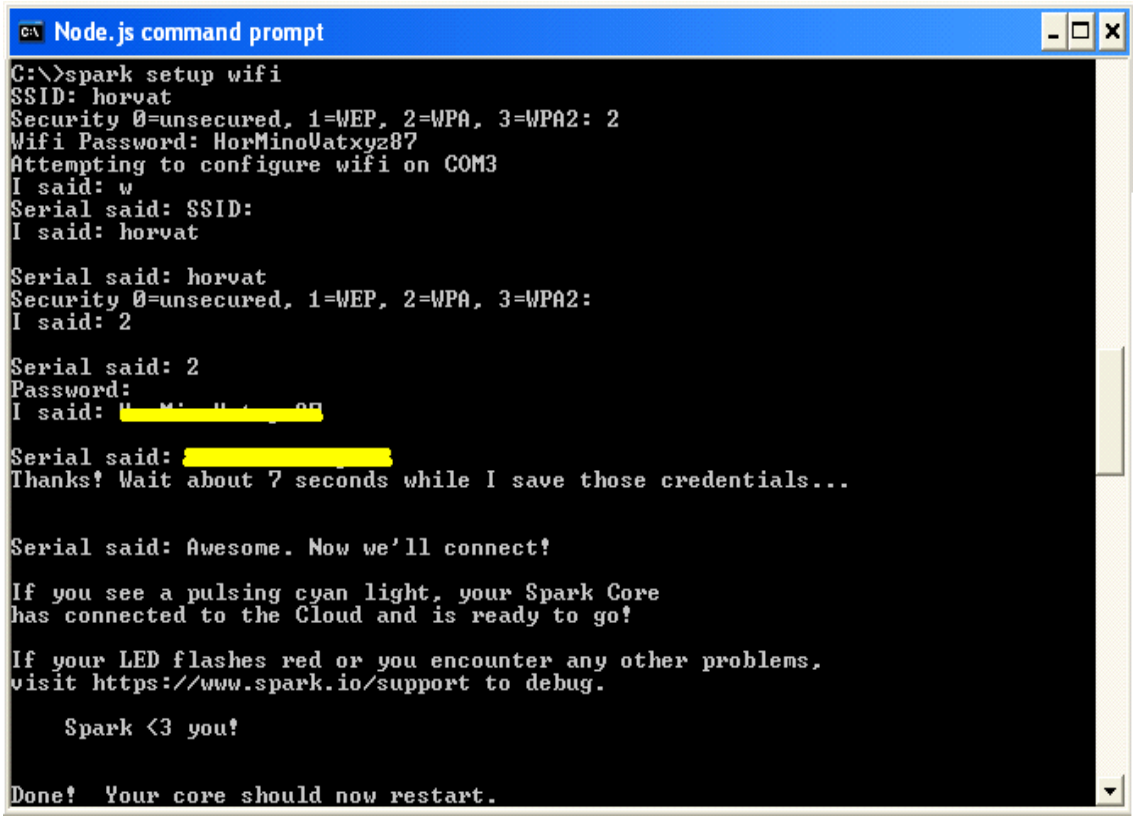
Slika 8. *Prijava u spark oblak i kreiranje korisničkog računa*



- Povezivanje sa usmjerivačem putem Wi-Fia pozivanjem naredbe `$ spark setup` wifi, gdje je potrebno upisati identifikator skupa usluga (engl. *SSID*), sadrži ga svaki BSS[XVII] ili ESS[XVIII] pomoću kojeg se identificira, to je obično ljudima lako čitljiv niz i često se naziva kao mrežno ime (engl. *network name*). U IBSS[XIX], SSID je izabran od strane klijentskog uređaja koji pokreće mrežu i emitiranje SSID. Izvodi se u pseudo slučajno odabranom redu od svih uređaja koji su članovi mreže, slika 9. prikazuje uspješno povezivanje sa Wi-Fiom.

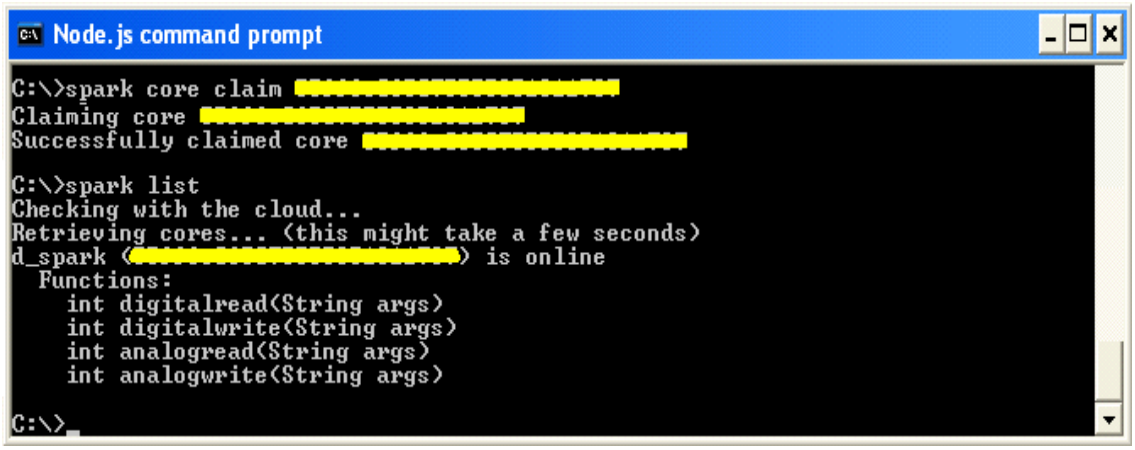
[XVII] BSS – Osnovni set usluga, osigurava osnovnu izgradnju bloka od 802.11 bežične lokalne mreže.
[XVIII] ESS – Prošireni set usluga, skup dvaju ili više međusobno bežično povezanih BSS-a koji djele isti SSID.
[XIX] IBSS – Neovisni BSS, sa 802.11 alternativno uspostavljena ad hoc mreža s klijentskim uređajima bez kontroliranja pristupne točke.

Slika 9. *Uspostava Wi-Fi konekcije sa usmjerivačem*



- Nadalje potrebno je napraviti potvrdu vlasništva (engl. *claim*) nad jezgrom, naredbom `$ spark cloud claim (core id)` gdje je potrebno upisati ID dobivene jezgre kako bi se izvršila naredba i završio cijeli proces, kao što je prikazano na slici 10. Nakon izvršenog procesa korisnik može slobodno vršiti željene interakcije, kao što je nadogradnja jezgre, prikaz dostupnih jezgri, slanje i primanje određenih zahtjeva te drugo [9].

Slika 10. *Uspješno potvrđivanje vlasništva nad jezgrom*



4.2. Programiranje mikrokontrolera

Programiranje samo po sebi može biti lako ili teško, ovisno od korisnika koji vrši programiranje. Pisanje koda za mikrokontroler vrši se putem web IDE dostupno na www.spark.io/build, pomoću Arduino/Wiring, C/C++ ili čak assembly programskih jezika, također mu se može pristupiti putem mobilnog uređaja.

Web IDE ili WIDE je web integrirano razvojno okruženje, također poznato kao cloud IDE, to je zapravo preglednik baziran na IDE-u koji omogućuje razvoj softvera ili razvoj weba. Pristupiti mu se može preko web preglednika kao što su google chrome, internet explorer i drugi, omogučujući prijenosno radno okruženje. Web IDE-i obično se sastoje od editora izvornog koda, automatizacijskog alata izgradnje i debuggera[XX], iako ne sadrže sve iste mogućnosti kao standardni ili desktop IDE-i, osnovne značajke kao npr. isticanje sintakse obično su prisutne.

Nadalje je prikazan kratki probni programski kod sa opisom koji se piše na web IDE-u, te koji ispisuje rezultat stanja izlaznog i ulaznog signala ovisno o odaslanom ili primljenom signalu, visok (engl. *HIGH*) što predstavlja logičku "1" i nizak (engl. *LOW*) kao logičku "0". Kod se jednostavno preuzme bežićno na samu jezgru.

```
#define ON 1                    //Definiranje log 1 kao ON.
#define OFF 0                   //Definiranje log 0 kao OFF.

//Određivanje out/input state kao integer.
int outputstate = 0;
int inputstate = 0;

//Unsigned integer 32-bit type.
uint32_t counter = 1;

void setup() {

    pinMode(D0, OUTPUT);   //Definiranje pina D0 kao izlaz.
    pinMode(D4, INPUT);    //Definiranje pina D4 kao ulaz.

//Uključivanje izlaza D0 (logička 1) kad se može pokrenuti
serijski terminal.
    digitalWrite(D7,HIGH);

//Pokretanje serijske veze na 9600 bita/s.
    Serial.begin(9600);
```

[XX] Debugger – Program za testiranje ispravnosti drugih programa, te pronalaženje i otklanjanje pogrešaka.

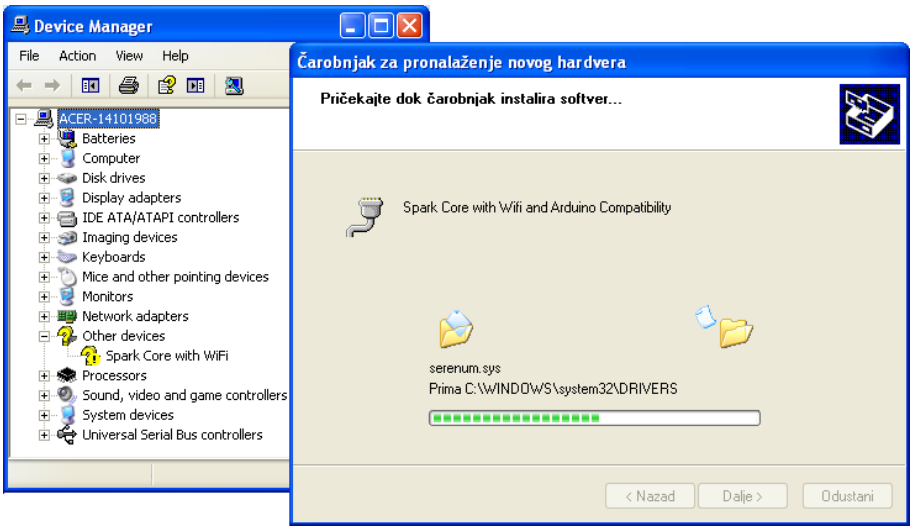
```
//Čekanje za pritisak tipke u terminalu.
    while(!Serial.available())

    SPARK_WLAN_Loop();     //Pozivanje pozadinskog rada loop.

//Isključivanje izlaza D0 (logička 0) i serijska komunikacija
započinje.
    digitalWrite(D7, LOW);
}

void loop()                //Izvršavanje beskonačne petlje.
{
//Ispisuje broj brojača te ga uvećava za +1.
    Serial.print(counter++);
    Serial.println(". Read / Write states over USB! ");
    Serial.print("\n");

    digitalWrite(D0, ON);       //Daje izlaz pina D0 kao log 1.

//Pohranjuje očitanu vrijednost sa D0.
    Outputstate = digitalRead(D0);
    Serial.print("Output: ");

//Ispisuje u terminalu pohranjenu vrijednost.
    Serial.print(outputstate);
    Serial.println(" = ON");

//Pohranjuje očitanu vrijednost sa D4.
    inputstate = digitalRead(D4);
    delay(1000);                      //Proces čeka 1 sekundu.
    Serial.print("Input: ");

//Provjerava upit ako je pohranjena vrijednost jednaka logičkoj
0, tada se izvršava sljedeće.
    if(inputstate == OFF){
      Serial.print("LOW");
    }

    else{
      Serial.print("HIGH");
    }
    delay(2000);
    Serial.println();
    digitalWrite(D0, OFF);
    outputstate = digitalRead(D0);
    Serial.print("Output: ");
    Serial.print(outputstate);
    Serial.println(" = OFF");

    inputstate = digitalRead(D4);
    delay(1000);
    Serial.print("Input: ");
```

```
//Provjerava upit ako je pohranjena vrijednost jednaka logičkoj
1, tada se izvršava sljedeće.
    if(inputstate == ON){
      Serial.print("HIGH");
    }

    else{
      Serial.print("LOW");
    }
    delay(3000);
    Serial.println("\n");
}
```

4.3. Testiranje ispravnosti ulazno-izlaznih signala serijskom usb vezom

Nakon napravljenog i preuzetog koda na mikrokontroler, potrebno je povezati hardver serijskim putem sa računalom. Kao i svaki novi hardver koji je prvi puta priključen serijski na računalo, zahtjeva od računala prepoznavanje te se sa upravljačkim programom hardvera (engl. *driver*) uspostavlja komunikacija između računala i hardvera, slika 11. prikazuje instalaciju drivera za spark hardver.

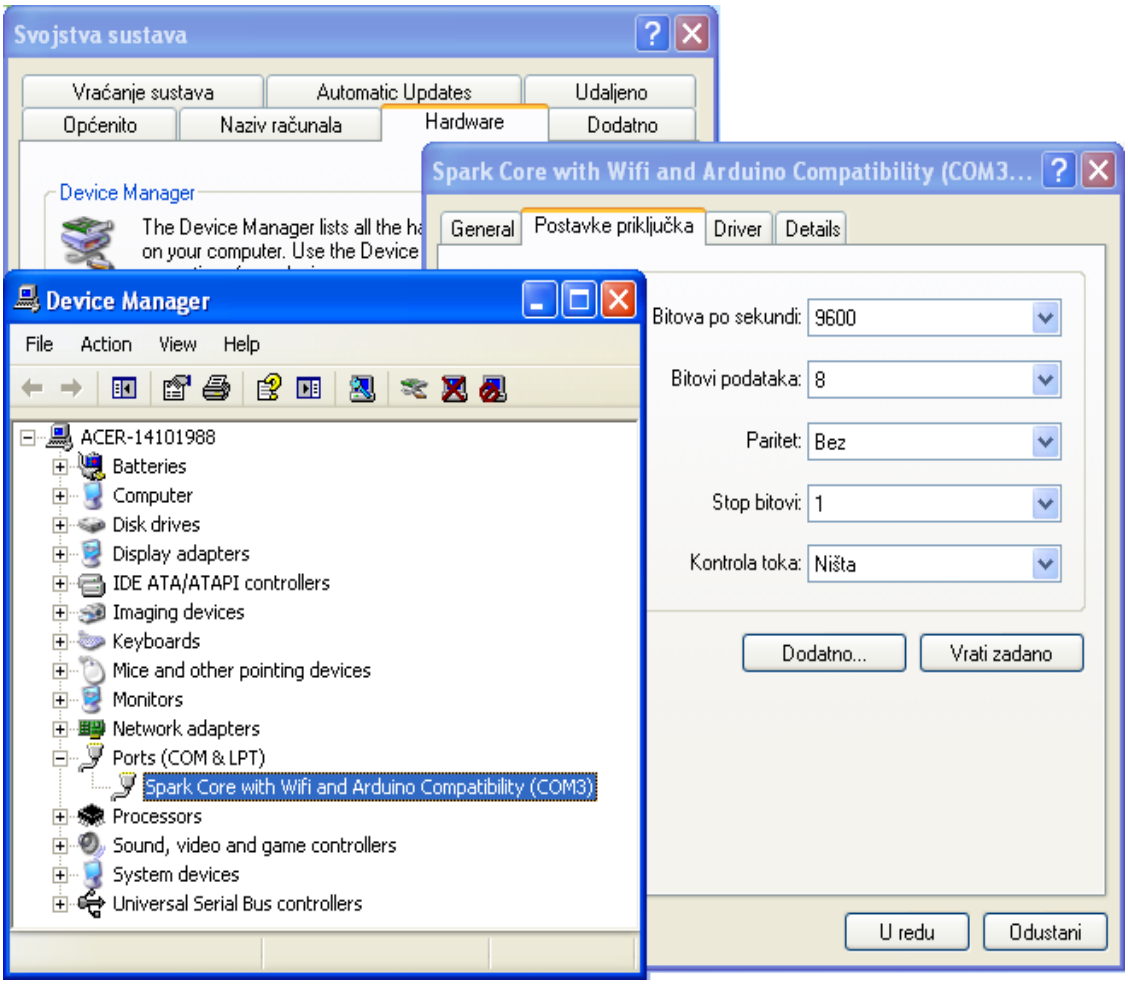
Slika 11. *Instalacija drivera za hardver Spark Core.*



Pregledom u upravitelj uređaja (engl. *Device Manager*) provjerimo ako je uspješno uspostavljena serijska veze putem usb-a, te utvrdimo na kojem se portu nalazi, u ovome slučaju to je port COM3[XXI]. Na slici 12. prikazana je uspješno uspostavljena serijska veza jezgre sa računalom putem usb-a.

[XXI] COM – Komunikacijski port.

Slika 12. *Uspješno uspostavljena serijska veza sa Spark jezgrom i postavke porta.*

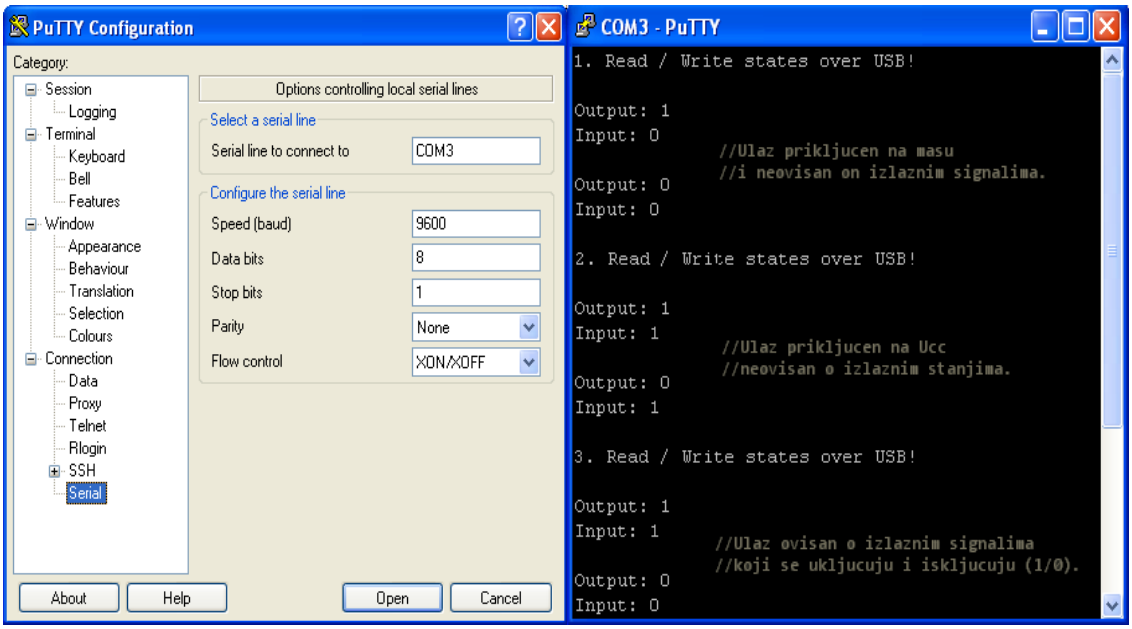


Nadalje za prikaz ispisa radnji mikrokontrolera u ovome projektu upotrebljen je program "PuTTY". To je besplatan terminalni emulator otvorenog koda, serijske konzole i mrežnog prijenosa podatkovnih aplikacija. Podržava nekoliko mrežnih protokola, uključujući SCP[XXII], SSH[XXIII], Telnet[XXIV], rlogin[XXV] i raw socket konekciju. U programu podesimo postavke serijske veze na port COM3, brzinu bita 9600 i pokrenemo serijski terminal, tek tada možemo vidjeti ispis rada mikrokontrolera koji se u ovome primjeru konstantno odrađuje, pošto se izvršava u beskonačnoj petlji

[XXII] SCP – Sredstvo sigurnog prijenosa računalnih datoteka između lokalnog hosta i udaljenog hosta ili između dva udaljena hostova (engl. *Secure Copy*).
[XXIII] SSH – Mrežni protokol koji korisnicima omogućuje uspostavu sigurnog komunikacijskog kanala između dva računala putem nesigurne računalne mreže (engl. *Secure Shell*).
[XXIV] Telnet – Mrežni protokol unutar IP grupe protokola koji se koristi na internetu ili u lokalnim mrežama.
[XXV] rlogin – Softverski alat za operativne sustave slične Unixu koji omogućava korisnicima da se prijave na drugi host preko mreže, komunicirajući putem TCP porta 513.

(engl. *loop*). Slika 13. prikazuje program PuTTY podešen na serijsku vezu sa usb-om i ispis rada mikrokontrolera na terminalu.

Slika 13. *Sa lijeve strane se vide postavke PuTTY-a za serijsku vezu, a desno rad mikrokontrolera sa ulazom i izlazom*

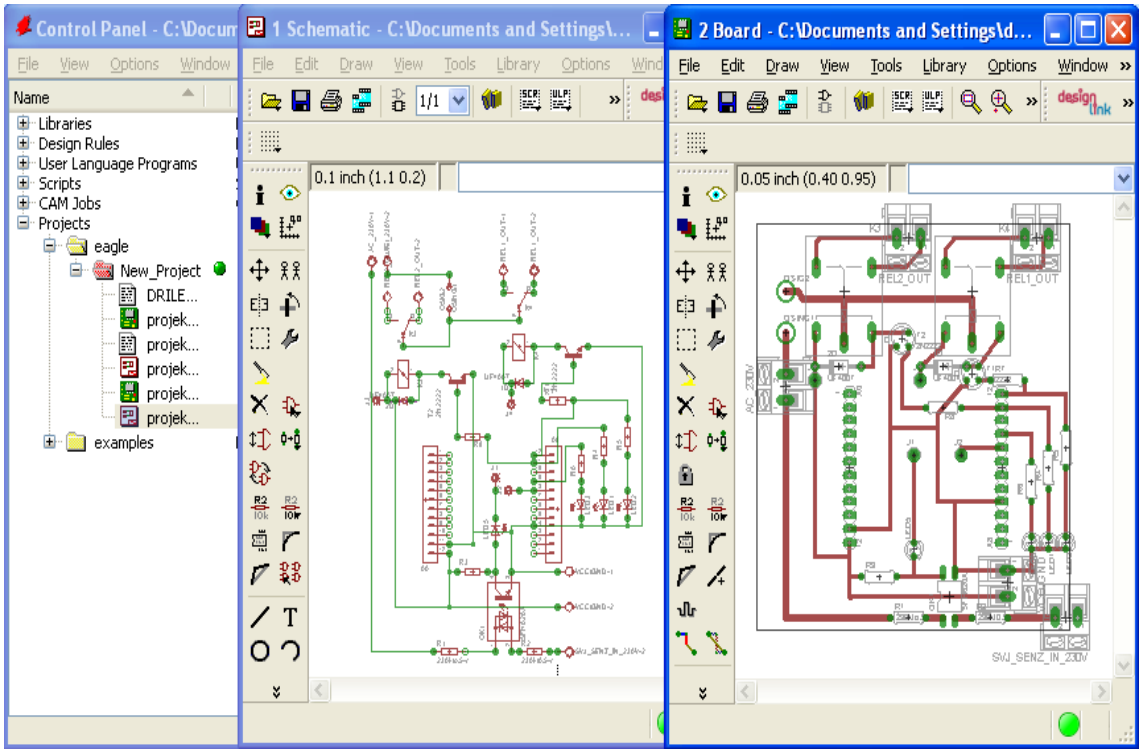


4.4. Izrada probnih shema sklopa za upravljanje strujnim krugom

Pošto nije moguće samo tako samu spark jezgru, tj. takozvani mozak projekta integrirati u strujni krug u kojem se koristi izmjenični napon od 230V, te dodatno istosmjerno napajanje samog sklopa od 5V, potrebno je osmisliti shemu u kojoj će sve navedeno biti i obuhvaćeno.

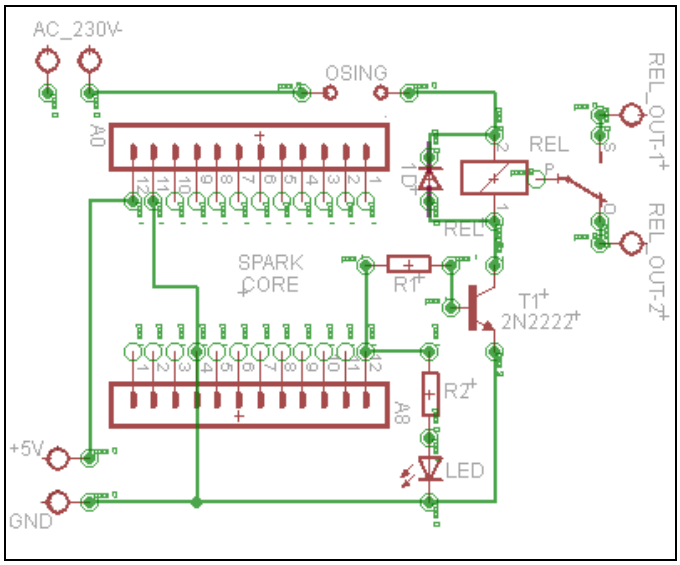
Dakle spark core drugim riječima mozak, biti će uključen u interakciju sa određenim strujnim krugom i elektroničkim uređajem, pomoću sklopa koji će biti izrađen prema napravljenoj i testiranoj shemi. Za izradu sheme koristi se "Eagle" program u prijevodu pod punim nazivom jednostavno primjenjiv uređivač grafičkog izgleda (engl. *Easily Applicable Graphical Layout Editor*), prikazan na slici 14. Za shemu je potrebno znati da će biti dva izvora napajanja, jedan izvor od 230 volti izmjenične struje koji će služiti za napajanje određenog uređaja, te 5 volti istosmjerne struje kao izvor napajanja spark jezgre.

Slika 14. *Eagle program*



Shema 1. je eksperimentalna i prikazuje jednostavni sklop koji ima jedan izlaz za upravljanje jednim uređajem.

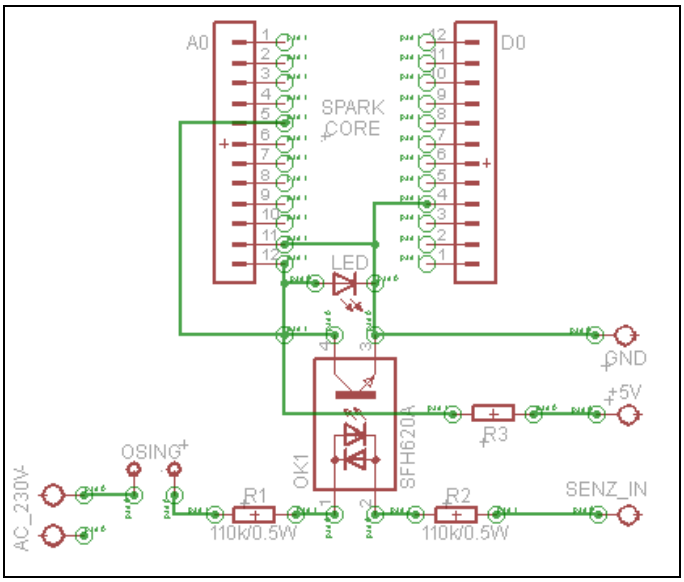
Shema 1. *Upravljanje sa jednim izlazom pomoću jezgre*



Shema 2. također služi u eksperimentalne svrhe te prikazuje jedan ulaz za očitavanje ispravnosti strujnog kruga. Svrha ovih shema je da se isproba funkcionalnost kako

izlaznog signala tako i ulaznog signala i to prvenstveno sa jednim uređajem te da se utvrde eventualne pogreške i nedostaci.

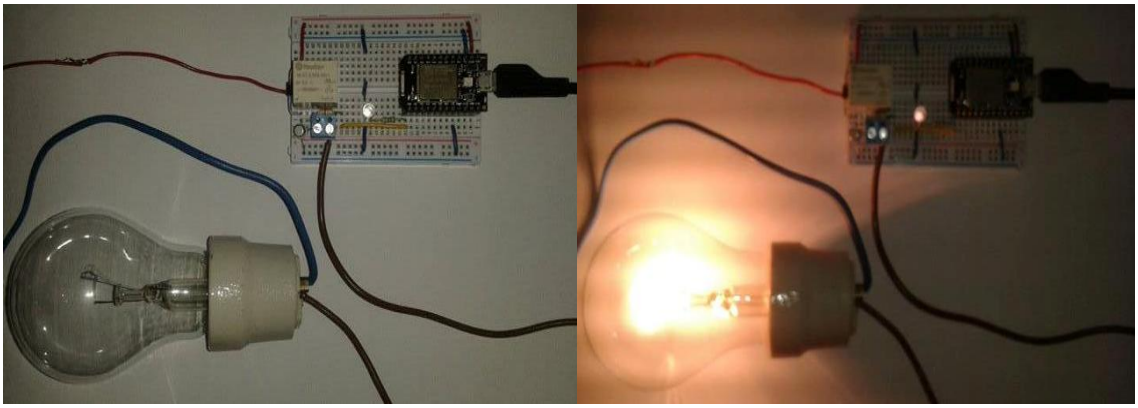
Shema 2. *Očitavanje ulaznog signala pomoću jezgre*



4.5. Testiranje probnih shema na eksperimentalnoj pločici

Po primjeru iz prve sheme izrađen je sklop na eksperimentalnoj pločici kojemu se doda spark core, te se vrši testiranje. Slika 15. prikazuje sklop koji aktivira relej pomoću izlaznih signala čije upravljanje vrši mikrokontroler sa prethodnim probnim kodom, koji se neprekidno izvršava.

Slika 15. *Upravljanje svjetlom sa izlaznim signalom, sa lijeve strane se vidi kada je izlazni signal 0, a desno kada je izlazni signal 1.*



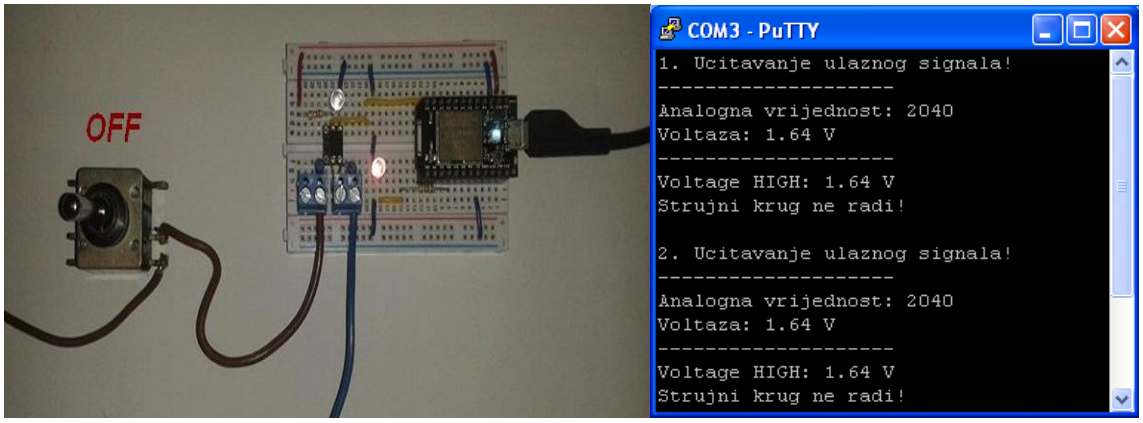
Za drugu shemu uzeto je u obzir da će se očitavanje ulaznog signala vršiti sa analognim ulazom mikrokontrolera, te je za to prilagođen programski kod koji se u ovome slučaju izvršava na mikrokontroleru. Prikaz programskog koda i objašnjenje kodnih linija nalazi se ispod teksta.

```
int inputValue;
float voltage;
uint32_t counter = 1;            //Unsigned integer 32-bit type.

void setup() {

    pinMode(D4, OUTPUT);        //Definiranje pina D4 kao izlaz.
    pinMode(A4, INPUT);         //Definiranje pina A4 kao ulaz.

//Uključivanje izlaza D7 kad se može pokrenuti serijski
terminal.
    digitalWrite(D7, HIGH);

//Pokretanje serijske veze na 9600 bita/s.
    Serial.begin(9600);

//Čekanje za pritisak tipke u terminalu.
    while(!Serial.available())

    SPARK_WLAN_Loop();     //Pokretanje loop petlje na sparku.

//Isključivanje izlaza D7 kada započinje loop.
    digitalWrite(D7, LOW);
}
void loop() {              //Izvršavanje beskonačne petlje.

//Ispis broja brojača povećanje za plus 1.
    Serial.print(counter++);

    Serial.println(". Ucitavanje ulaznog signala!");
    Serial.println("--------------------");

//Učitavanje ulaznog signala na analognom A4 pinu.
    inputValue = analogRead(A4);
    Serial.print("Analogna vrijednost: ");

//Ispis pohranjene ulazne analogne vrijednosti.
    Serial.println(inputValue);
    delay(1000);

//Pretvorba ulazne analogne vrijednosti u voltažnu vrijednost
(raspona 0V – 3.3V).
    voltage = inputValue * (3.3 / 4096.0);

    Serial.print("Voltaza: ");
    Serial.print(voltage);                  //Ispis voltaže.
```

```
    Serial.println(" V");
    Serial.println("--------------------");
    delay(2000);

//Upit ako je voltaža >=1.2 izvršava se sljedeća radnja.
    if (voltage >= 1.2){

//Postavljanje D4 izlaza u log 1.
      digitalWrite(D4, HIGH);
      Serial.print("Voltage HIGH: ");
      Serial.print(voltage);
      Serial.println(" V");
      Serial.println("Strujni krug ne radi!");
    }

    else {

//Postavljanje izlaza D4 u log 0.
      digitalWrite(D4, LOW);
      Serial.print("Voltage LOW: ");
      Serial.print(voltage);
      Serial.println(" V");
      Serial.println("Strujni krug radi.");
    }

//Kašnjenje za 4 sekunde.
    delay(4000);
    Serial.println();
}
```

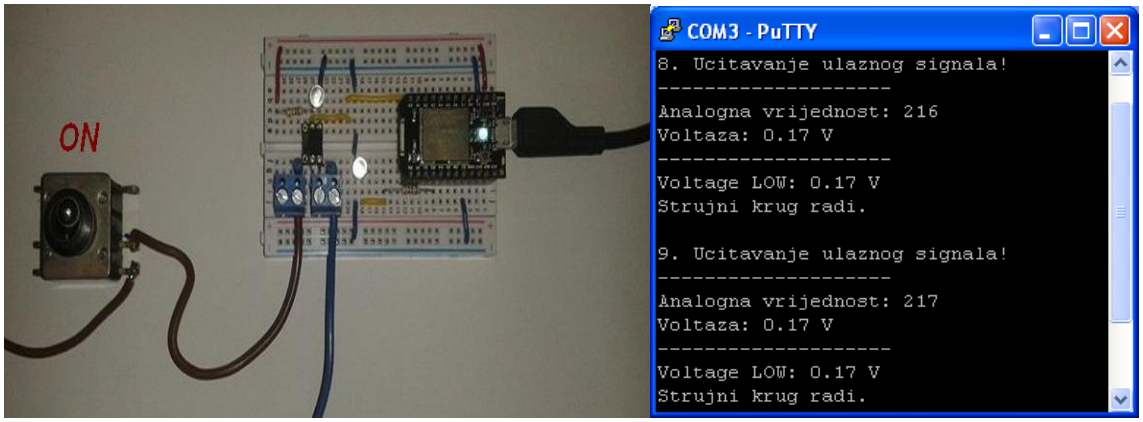
Nakon izrađenog koda prema shemi 2 izrađen je sklop na eksperimentalnoj pločici za provjeru ispravnosti strujnog kruga koji vrši očitavanje ulaznog signala ovisno o protoku struje u strujnom krugu. Slika 16. prikazuje sklop kada je strujni krug isključen, prekinut ili ne radi iz nekih određenih razloga, te prikaz u programu PuTTY ispisa rada mikrokontrolera koji prati ulazne signale.

Slika 16. *Eksperimentalni sklop sa očitavanjem i ispisom ulaznih signala isključenog strujnog kruga*



Na slici 17. također se nalazi identičan sklop kao u prethodnoj slici samo što se ovdje radi o uključenom strujnom krugu koji ispravno radi i sadrži prikaz ispisa radnji mikrokontrolera ulaznog stanja ovisno o aktivnom stanju strujnog kruga.

Slika 17. *Eksperimentalni sklop sa očitavanjem i ispisom ulaznih signala uključenog strujnog kruga*

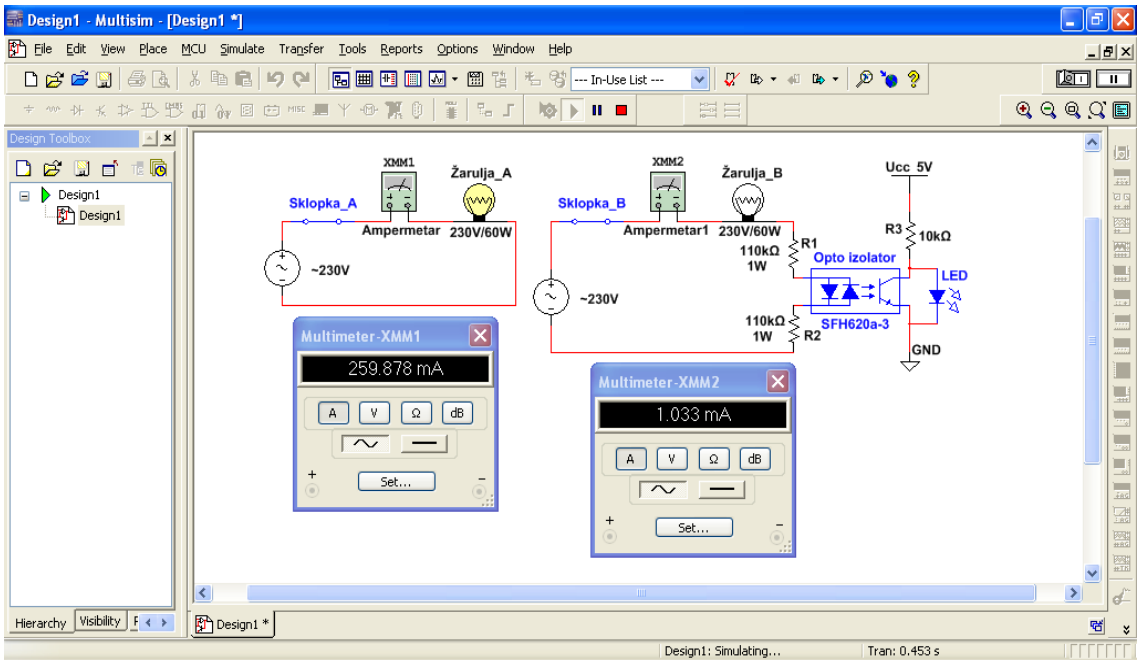


4.6. Problem sheme 2 i pronalazak rješenja

Kako se vršilo testiranje druge sheme došlo se do problema kada se u strujni krug gdje se ispitivala ispravnost samog strujnog kruga dodalo neko trošilo, konkretno u ovome slučaju žarulja od 60W/230V. Razlog je bio utvrđen provjerom putem programa "NI Multisim" koji se korisiti za prikazivanje i simulaciju elektroničkih shema, kako je

prikazano na slici 18. gdje se ustanovilo da je potrebna struja nedovoljna, što nam i govori sam izračunu prema formuli za disipaciju snage P = U * I odakle se izračuna jakost struje koja iznosi I=260mA za žarulju od 60W kako bi mogla ispravno raditi, a za opto sprežnik[XXVI] maksimalna jakost struje iznosi I=60mA.

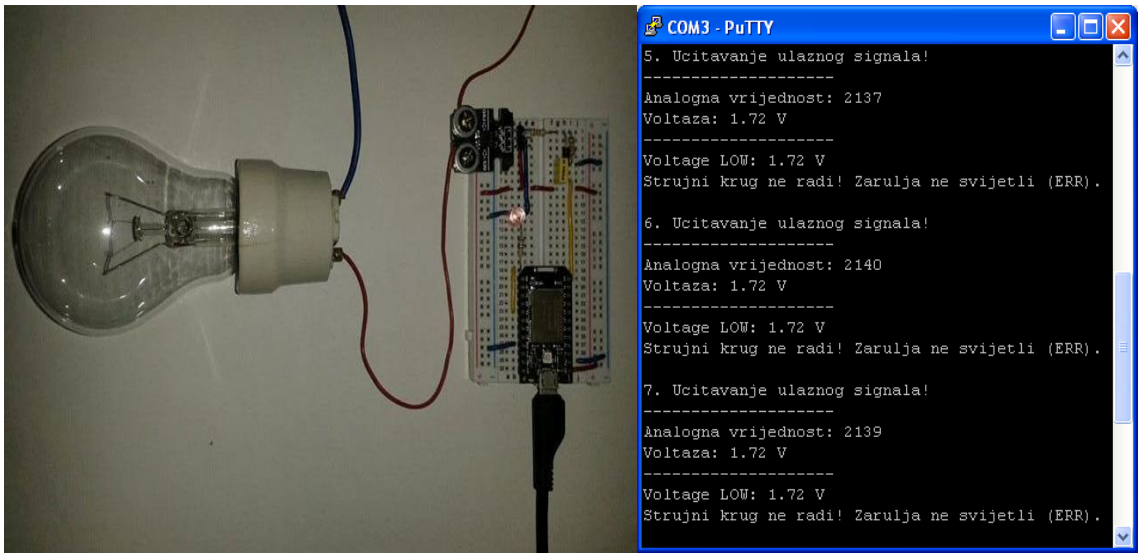
Slika 18. *Multisim sa prikazom jakosti struje kroz žarulju i prikaz iznosa struje prema shemi 2*



Dakle kada se prema shemi 2 doda u strujni krug otpornik sa kojim se spriječava preopterećenje i smanji jakost struje za opto sprežnik kako bi ispravno radio, time se i oduzme potrebna struja za ispravan rad žarulje, drugim riječima žarulja neće svijetliti. Ovakva provjera ispravnosti strujnog kruga idealna je ukoliko se koristi samo za provjeru rada ili prekida strujnog kruga, tako da nije ovisna o trošilu čime se omogućuje nadzor nad samim strujnim krugom. Stoga kao senzor ispravnosti strujnog kruga upotrebljen je ACS712[XXVII] temeljen na Hallovom efektu[XXVIII]. Slika 19. prikazuje sklop na eksperimentalnoj pločici sa neispravnom žaruljom i radom mikrokontrolera koji serijskom vezom ispisuje svoj rad, kako bi se utvrdila ovisnost očitane vrijednosti sa neispravnošću žarulje.

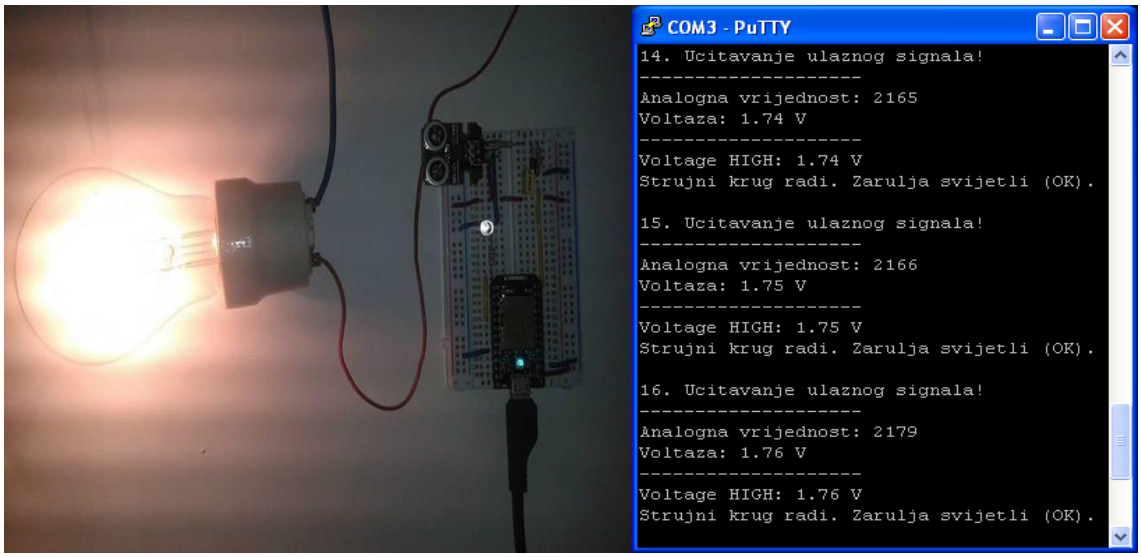
[XXVI] Opto sprežnik – Komponenta koja prenosi signale između dva izolirana strujna kruga pomoću svjetlosti.
[XXVII] ACS172 – Modul strujnog senzora temeljen na Hallovom efektu.
[XXVIII] Hallov efekt – Opisuje pojavu električnog napona u strujnom vodiču, koji se nalazi u stabilnom magnetskom polju.

Slika 19. *Rad neispravne žarulje na eksperimentalnom sklopu sa ispisom rada mikrokontrolera*



Na slici 20. prikazuje jednak sklop sa ispravnom žaruljom i također vidljivim ispisom rada mikrokontrolera, tj. ovisnost ulaznog stanja u odnosu na ispravan rad žarulje.

Slika 20. *Rad neispravne žarulje na eksperimentalnom sklopu sa ispisom rada mikrokontrolera*



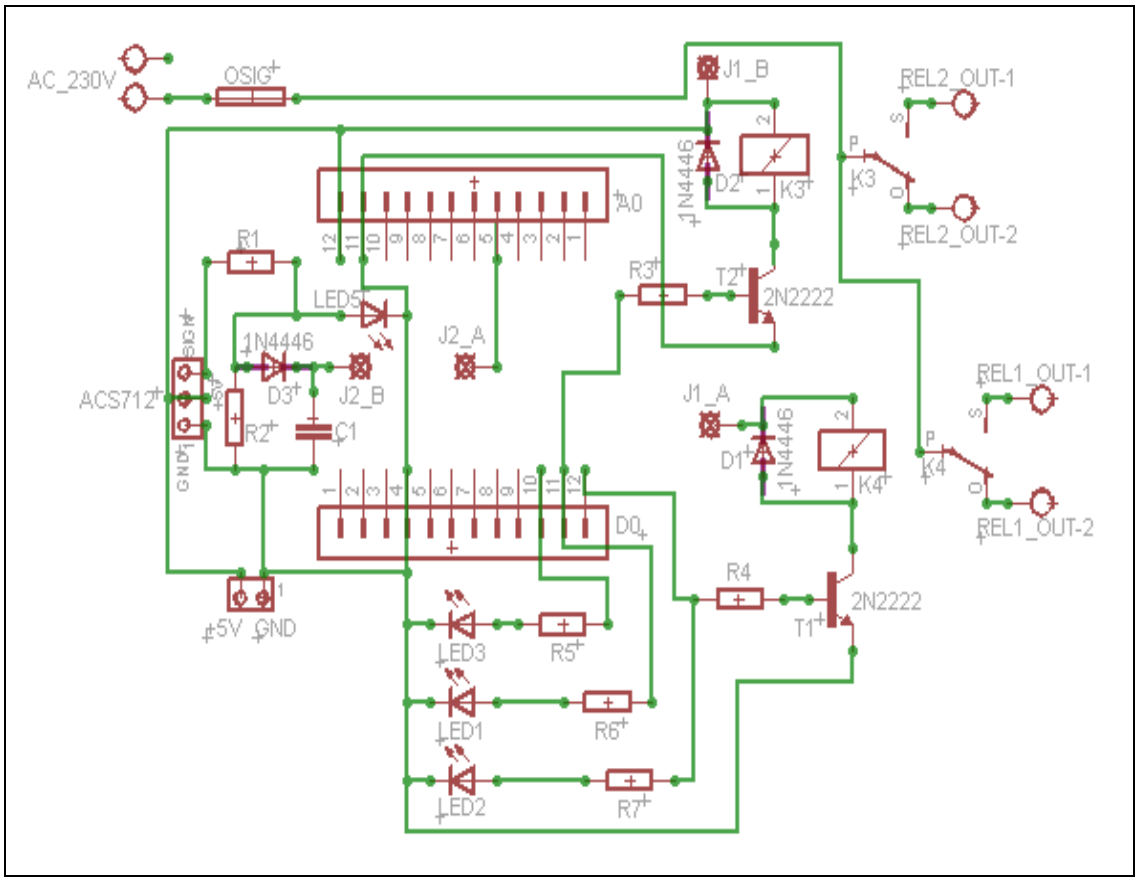
Potrebno je napomenuti kako treba precizno odrediti vrijednost (može biti učitana analogna ili pretvorena u volte) koja predstavlja ispravnost i neispravnost žarulje kako bi mikrokontroler mogao obavljati određenu operaciju. Iz specifikacija za ACS712 vidljivo je da je odnos 100mV/1A, što znači da za žarulju koja koristi 260mA promjena u voltaži iznosi 26mV. Tako je i iz navedenog primjera sa slike vidljivo da očitanje iz neispravnog strujnog kruga iznosi od 1.71V do 1.73V (variranje analogne vrijednosti od

2135 do 2153) dok očitanje iz ispravnog strujnog kruga iznosi od 1.74V do 1.76V (variranje analogne vrijednosti od 2154 na više). Testiranjem ispravnosti žarulje pomoću ACS712 utvrđeno je kako je praktičnije i jednostavnije određivanje vrijednosti koje predstavljaju rad strujnog kruga ukoliko žarulja ili neki uređaj zahtjeva više struje za rad. Tako npr. ako uzmemo auto svijetlo potrošnje 55W/12V istosmjerne struje, gdje jakost struje iznosi 4.583A, što znači da će se pomoću ACS712 zabilježiti znatna promjenu od 458.33mV.

4.7. Izrada glavne sheme cijelog sklopa

Shema 3. koja služi kao završna, su zapravo prva i druga ispravljena shema uparene u jednu sa nekim izmjenama i dodacima kao osigurač iz sigurnosnih razloga, više izlaza potrebnih za upravljanje sa više uređaja, od čega su dva izlaza povezana sa relejima, koji će kasnije poslužiti za konkretni prikaz ovoga projekta.

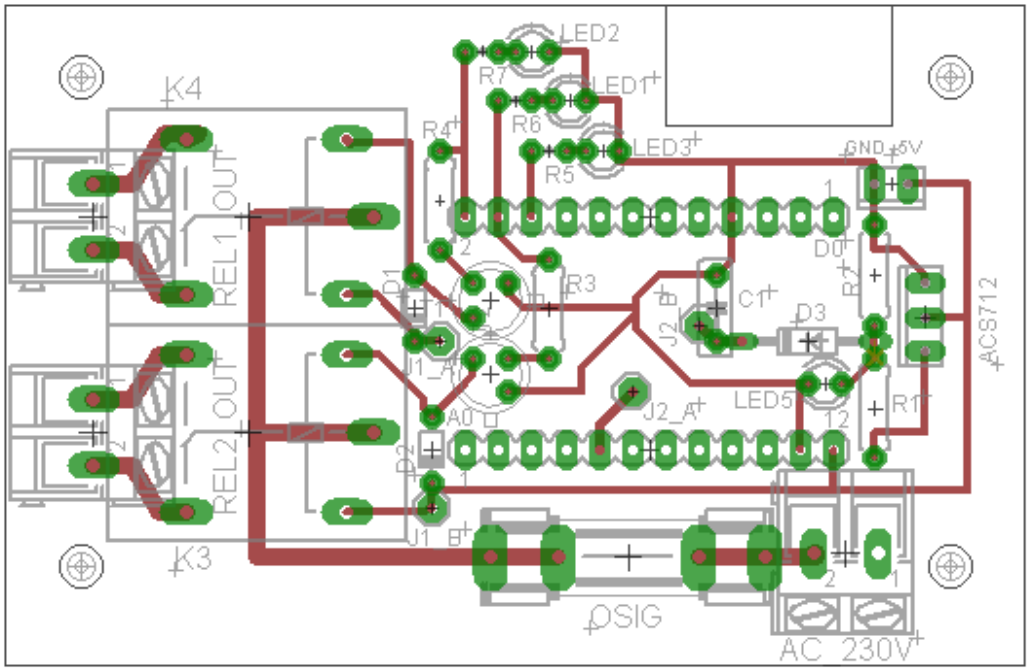
Shema 3. *Završna shema*



4.8. Izrada glavnog sklopa

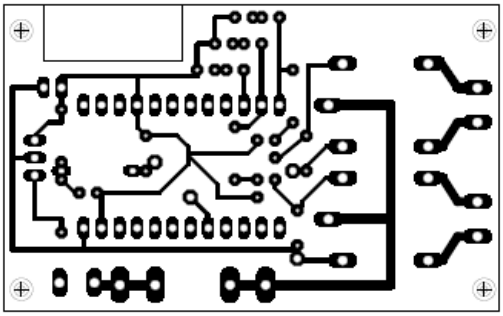
Nakon napravljene sheme i eksperimentalne provjere ispravnosti, potrebno je načiniti stvarni sklop za korištenje i u praktičnom djelu. Tako je i sa programom Eagle izrađen izgled i raspored elemenata za tiskanu pločicu. Na slici 21. prikazan je napravljeni raspored elemenata na elektroničkoj pločici pomoću programu Eagle [10].

Slika 21. *Raspored elemenata na elektroničkoj pločici*

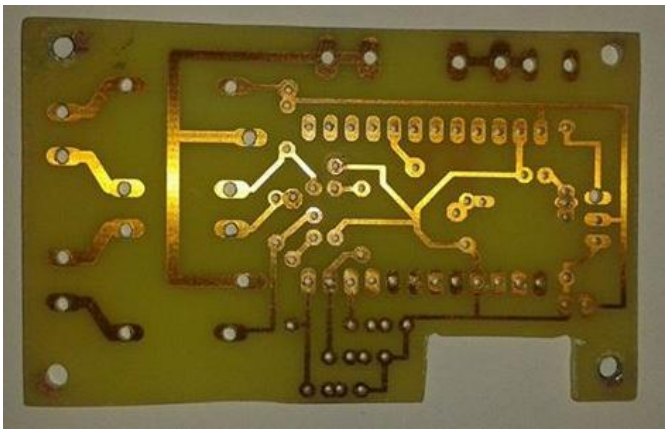


U daljnjem postupku kao što prikazuje slika 22. u mjerilu M 1:1 izrađen je otisak vodova koji će se prenjeti na prozirnu foliju kako bi se mogla razviti elektronička pločica foto postupkom, a na slici 23. može se vidjeti gotova elektronička pločica sa razvijenim vodovima.

Slika 22. *Otisak vodova za elektroničku pločicu, M 1:1*

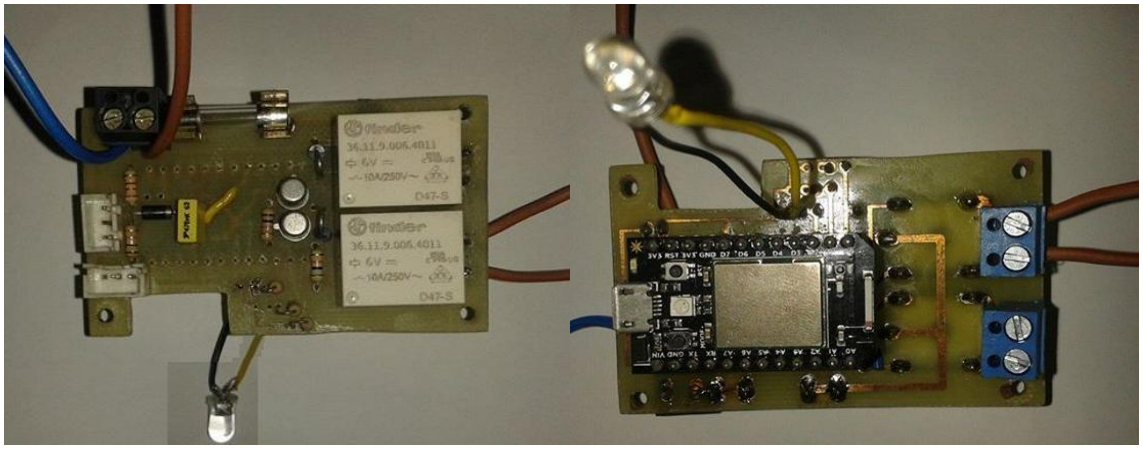


Slika 23. *Elektronička pločica sa razvijenim vodovima*



Nadalje slijedi ugrađivanje elektroničkih komponenata zajedno sa Spark jezgrom kako bi se sve zajedno moglo integrirati u strujni krug. Gotova pločica izrađenog sklopa sa dodanim sparkom prikazana je na slici 24.

Slika 24. *Izrađeni sklop sa Spark jezgrom*



4.9. Cjeloviti programski kod za mikrokontroler

Prikaz i opis cjelovitog napisanog programskog koda koji izvršava mikrokontroler na jezgri i koji se upisuje na web IDE nalazi se ispod teksta.

```
#define ON 1                        //definiranje ON kao log 1.
#define OFF 0                       //definiranje OFF kao log 0.

//Postavljanje brojača sa početnom vrijednošću 1.
uint32_t counter1 = 1, counter2 = 1, counter3 = 1;

int lightState, doorState, zaruljaState;
int inputValue;
```

```
float voltage;
void setup() {

//Pozivanje spark funkcije na pritisak određene tipke.
    Spark.function("RoomLight", test1Function);
    Spark.function("GarageDoor", test2Function);
    Spark.function("Zarulja", zaruljaFunction);

//Pozivanje spark vriable ovisno o pritisku određene tipke.
    Spark.variable("lightstate", &lightState, INT);
    Spark.variable("doorstate", &doorState, INT);
    Spark.variable("zaruljastate", &zaruljaState, INT);

    pinMode(A4, INPUT);    //Definiranje pina A4 kao ulaz.
    pinMode(D0, OUTPUT);   //Definiranje pina D0 kao izlaz.
    pinMode(D1, OUTPUT);   //Definiranje pina D1 kao izlaz.
    pinMode(D2, OUTPUT);   //Definiranje pina D2 kao izlaz.
    pinMode(D7, OUTPUT);   //Definiranje pina D7 kao izlaz.

//Postavljanje digitalnih izlaza u log 0.
    digitalWrite(D0, LOW);
    digitalWrite(D1, LOW);
    digitalWrite(D2, LOW);
    digitalWrite(D7, LOW);

//Pokretanje serijske veze na 9600 bita/s.
    Serial.begin(9600);

//Čekanje za pritisak tipke u terminalu.
    while(!Serial.available())

//Pokretanje sparkove beskonačne petlje.
    SPARK_WLAN_Loop();
}
void loop()
{}

//Izvršavanje test1Function koja koristi jedan izlaz (D0).
int testFunction(String outputPin) {

    Serial.println();
    Serial.println("---------------");
    Serial.print(counter1++);
    Serial.println(". Pritisnuto.");

    if(lightState == OFF)
    {
      lightState = ON;
      digitalWrite(D0, lightState);
      Serial.print("Svjetlo: ");
      Serial.println(ON);
    }
```

```
    else {
      lightState = OFF;
      digitalWrite(D0, lightState);
      Serial.print("Svjetlo: ");
      Serial.println(OFF);
    }

//Naredba se provjerava u aplikaciji da potvrdi ispravnost.
    return 200;
    Serial.println("---------------");
}

//Izvršavanje test2Function koja koristi D1 izlaz.
int test2Function(String outputPin) {

    Serial.println();
    Serial.print("---------------");
    Serial.print(counter2++);
    Serial.println(". Aktivna.");

    if(doorState == OFF)
    {
      doorState = ON;
      digitalWrite(D1, doorState);
      Serial.print("Garazna vrata: ");
      Serial.println(ON);
    }

    else {
      doorState = OFF;
      digitalWrite(D1, doorState);
      Serial.print("Garazna vrata: ");
      Serial.println(OFF);
    }

    return 200;
    Serial.println("---------------");
}

//Izvršavanje zaruljaFunction koja učitava ulaznos stanje pina
A4.
int zaruljaFunction(String inputPin) {

    Serial.println();
    Serial.println("--------------------");
    Serial.print(counter3++);
    Serial.println(". Ucitavanje ulaznog signala!");
    Serial.println("--------------------");

//Učitavanje ulaznog signala na analognom pinu i pohranjivanje u
integer vrijednost (inputValue).
    inputValue = analogRead(A4);
    Serial.print("Analogna vrijednost: ");
```

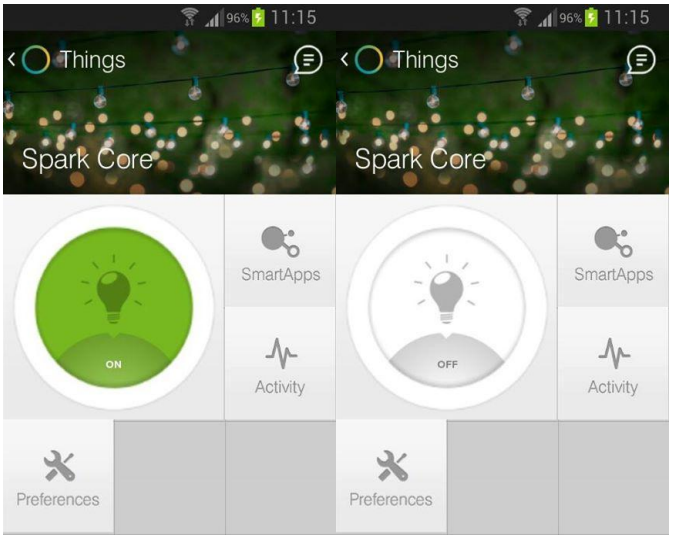
```
//Ispis pohranjene ulazne analogne vrijednosti.
      Serial.println(inputValue);

//Pretvorba ulazne analogne vrijednosti u voltažni iznos raspona
(0V-3.3V).
    voltage = inputValue * (3.3 / 4096.0);
    Serial.print("Voltaza: ");
    Serial.print(voltage);                  //Ispis voltaže.
    Serial.println(" V");
    Serial.println("--------------------");

//Upit ako je voltaža >=1.735 izvršuje se sljedeća radnja.
    if (voltage >= 1.735) {

//Postavljanje D7 izlaza u log 1.
       digitalWrite(D7, HIGH);
       Serial.print("Voltage HIGH: ");
       Serial.print(voltage);
       Serial.println(" V");
       Serial.println("Strujni krug ne radi!");

//Promjena stanja žarulje u log 0 (OFF).
       zaruljaState = OFF;
    }
    else {

//Postavljanje izlaza D7 u log 0.
       digitalWrite(D7, LOW);
       Serial.print("Voltage LOW: ");
       Serial.print(voltage);
       Serial.println(" V");
       Serial.println("Strujni krug radi.");
       zaruljaState = ON;
    }
    return 200;
    Serial.println();
}
```

4.10. Izrada probne web aplikacije

Da bi se moglo upravljati mikrokontrolerom putem wifia kako bi mogli vidjeti očekivani rezultat, potrebna je neka vrsta sučelja za upravljanje poput jednostavne aplikacije. Za primjer kako bi trenutno zamišljena aplikacija funkcionirala, te vizualno izgledala, postoje nekoliko vrsta aplikacija koje omogućavaju komunikaciju sa pametnim uređajima. Jedna takva aplikacija je "SmartThings Mobile", praktičnog

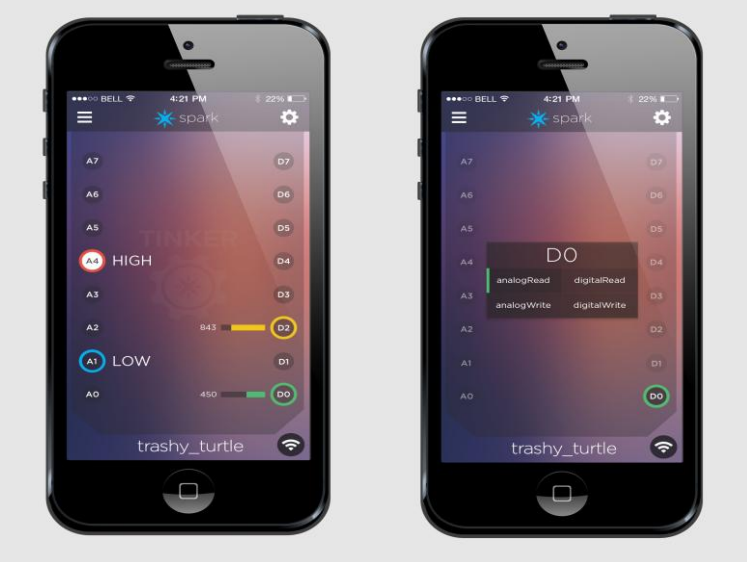
vizualnog izgleda prikazana na slici 25. pri čemu je potrebno imati hub[XXIX] dotičnog proizvođača kako bi se mogla uspostaviti komunikacija.

Slika 25. *SmartThings aplikacija*



Također sam Spark Core ima i svoju mobilnu "Spark" aplikaciju za upravljanje, iako nije nekog posebnog vizualnog izgleda, pošto je namjenjena da prikazuje sa kojim se izlazom ili ulazom upravlja što je vidljivo na slici 26.

Slika 26. *Spark aplikacija*

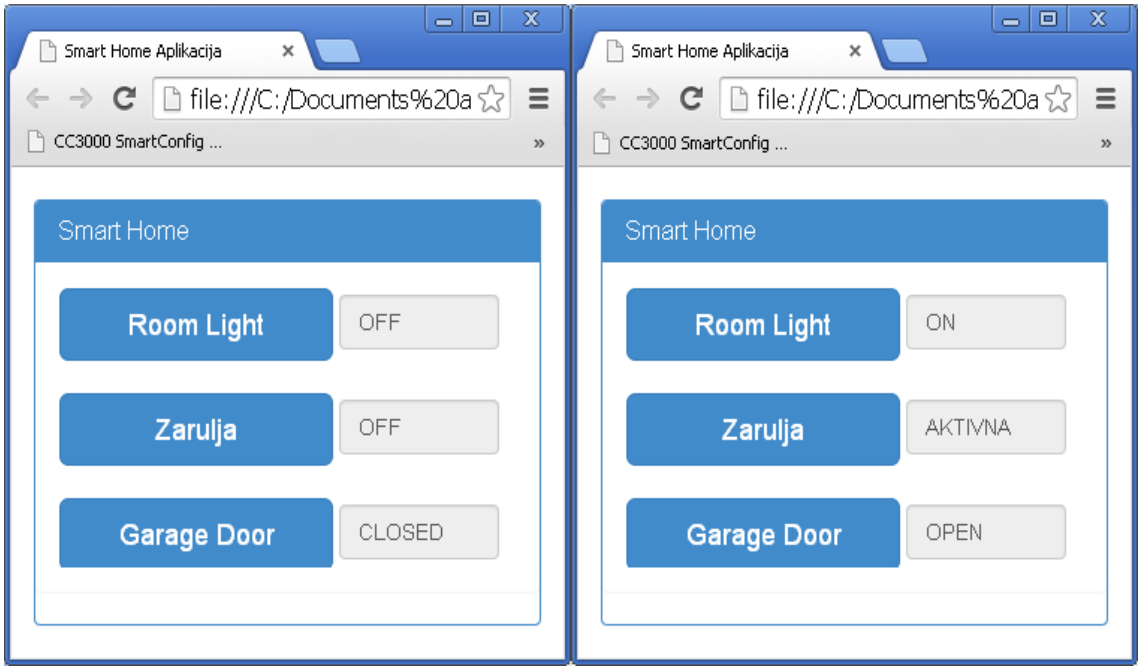


[11]

[XXIX] Hub – Koncentrator, sklop koji služi kao sredstvo preko kojeg se usredotočuje, odnosno preko kojeg se priključuje veći broj računala koja su povezana u istu računalnu mrežu.

Na taj se način pomoću Spark aplikacije lako može testirati interakcija između softvera i hardvera. Dakle besplatna Spark aplikacija preuzme se sa interneta i vrše se probna testiranja sa hardverom, te kad je provjerena njihova ispravnost, potrebna je realizacija vlastite aplikacije. Nakon viđenih primjera raznih upravljačkih aplikacija, dolazi se do jednostavnog zaključka za aplikaciju sa tipkama i prikaz stanja istih. Slika 27. prikazuje web aplikaciju koja sadrži tri tipke (Room Light, Žarulja i Garage Door) sa njihovim stanjima (OFF/ON, OFF/AKTIVNA i CLOSED/OPEN) napravljena pomoću HTML[XXX] programskog jezika sa CSS[XXXI] i JavaScript[XXXII] dodacima, te se pokreće na webu.

Slika 27. *Web aplikacija sa tipkama i njihovim stanjima*



Pošto je danas moguće naći svakakve primjere izrađenih web aplikacija pomoću html programskog jezika sa funkcionalnim tipkama koje koriste javascript, te su otvorenoga koda i dostupne za vlastitu upotrebu, pa tako i u ovome projektu će biti iskorišten jedan od dostupnih htmla uz pojedine preinake za komunikaciju sa hardverom. Orginalna verzija web aplikacije nalazi se na web stranici "https://github.com/technobly/Remote-

[XXX] HTML – Prezentacijski jezik za izradu web stranica (engl. *HyperText Markup Language*).
[XXXI] CSS – Stilski jezik, koji se upotrebljava za opis prezentacije dokumenata napisanog pomoću html jezika (engl. *Cascading Style Sheets*).
[XXXII] JavaScript – Skriptni programski jezik, koji se izvršava u web pregledniku na strani korisnika.

Spark/tree/master/JUSTGAUGE%20DEMO" [12]. Ispod teksta nalazi se izmjenjeni kod web aplikacije, koja se koristi u ovome projektu [13].

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Smart Home
    </title>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">
  </head>
  <body>
   <script type="text/javascript"
src="http://codeorigin.jquery.com/jquery-2.0.3.min.js"></script>
   <script type="text/javascript"
src="js/bootstrap.min.js"></script>
   <script>
    $(document).ready(function() {
        var statusTimer = null;
        var timer1, timer2, timer3;
        var baseURL = "https://api.spark.io/v1/devices/";
        var accessToken =
"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
        var coreID = "xxxxxxxxxxxxxxxxxxxxxxxx";

//Naziv aplikacije
        var appHeading = "Smart Home";

//Funkcija 1
        var funcKey1 = "RoomLight";
        var label1 = "Room Light"; //naziv gumba 1
//Varijabla 1
        var varKey1 = "lightstate";
        var var1onLabel = "ON";
        var var1offLabel = "OFF";
        var var1onState = 1; //stanje varijable on(1)/off(0)
        var var1offState = 0;
        var refresh1 = 1000; //vrijeme refrešanja varijable

//Funkcija 2
        var funcKey2 = "zarulja";
        var label2 = "Zarulja";
//Varijabla 2
        var varKey2 = "zaruljastate";
        var var2onLabel = "AKTIV";
        var var2offLabel = "OFF";
        var var2onState = 1;
        var var2offState = 0;
        var refresh2 = 1000;
```

```
//Funkcija 3
        var funcKey3 = "GarageDoor";
        var label3 = "Garage Door";
//Varijabla 3
        var varKey3 = "doorstate";
        var var3onLabel = "OPEN";
        var var3offLabel = "CLOSED";
        var var3onState = 1;
        var var3offState = 0;
        var refresh3 = 1000;

//Ažuriranje naslova aplikacije
     $("#app-heading").html(appHeading);
//Zaobilaženje stilova tipki
     $(".btn-lg").css({"width":"60%"});

//Ažuriranje naziva tipke kodnim definicijama ili njihovo
sakrivanje u neaktivnom stanju
     (label1) ? $("#button-1").html(label1) : $("#button-
1").hide();
     (label2) ? $("#button-2").html(label2) : $("#button-
2").hide();
     (label3) ? $("#button-3").html(label3) : $("#button-
3").hide();

//Zaobilaženje stila tekstualnog unosa
     $(".form-control").css({"display":"inline","width":"35%"});
     //Sakrivanje varijable tekstualnog polja ako se ne koristi
     if(refresh1 === 0) $("#var-val-1").hide();
     if(refresh2 === 0) $("#var-val-2").hide();
     if(refresh3 === 0) $("#var-val-3").hide();

//Automatsko osvježavanje(auto-refresh)
//Uključi/isključi varijablu refresh ako su definirana
osvježavanja.
     if(refresh1) {
        $("#get-var-1").attr("disabled", "disabled");
        timer1 = setInterval(function () {
          getVariable1();
        }, refresh1);
     }
     if(refresh2) {
        $("#get-var-2").attr("disabled", "disabled");
        timer2 = setInterval(function () {
          getVariable2();
        }, refresh2);
     }
     if(refresh3) {
        $("#get-var-3").attr("disabled", "disabled");
        timer3 = setInterval(function () {
          getVariable3();
        }, refresh3);
```

```
      }
//Upozorenja(Alerts)
      $("#info-alert").alert();
      $("#info-alert").affix();

//Metode
//Za uspješnu metodu
      function onMethodSuccess() {
         alert = $("#info-alert");
         //alert.text("Success!").removeClass("alert-
danger").addClass("alert-success");
         if(statusTimer != null) {
           clearTimeout(statusTimer);
           alert.hide();
           setTimeout(function() {
             alert.show();
             statusTimer = setTimeout(function() {
               statusTimer = null;
               alert.hide();
             }, 1750);
           }, 250);
         }
         else {
           alert.show();
           statusTimer = setTimeout(function() {
             statusTimer = null;
             alert.hide();
           }, 2000);
         }
      }
//Za neuspješnu metodu
      function onMethodFailure(data) {
         alert = $("#info-alert");
         //alert.text((data)?"Error!
"+data:"Error!").removeClass("alert-success").addClass("alert-
danger");
         if(statusTimer != null) {
           clearTimeout(statusTimer);
           alert.hide();
           setTimeout(function() {
             alert.show();
             statusTimer = setTimeout(function() {
               statusTimer = null;
               alert.hide();
             }, 1750);
           }, 250);
         }
         else {
           alert.show();
           statusTimer = setTimeout(function() {
             statusTimer = null;
             alert.hide();
           }, 2000);
```

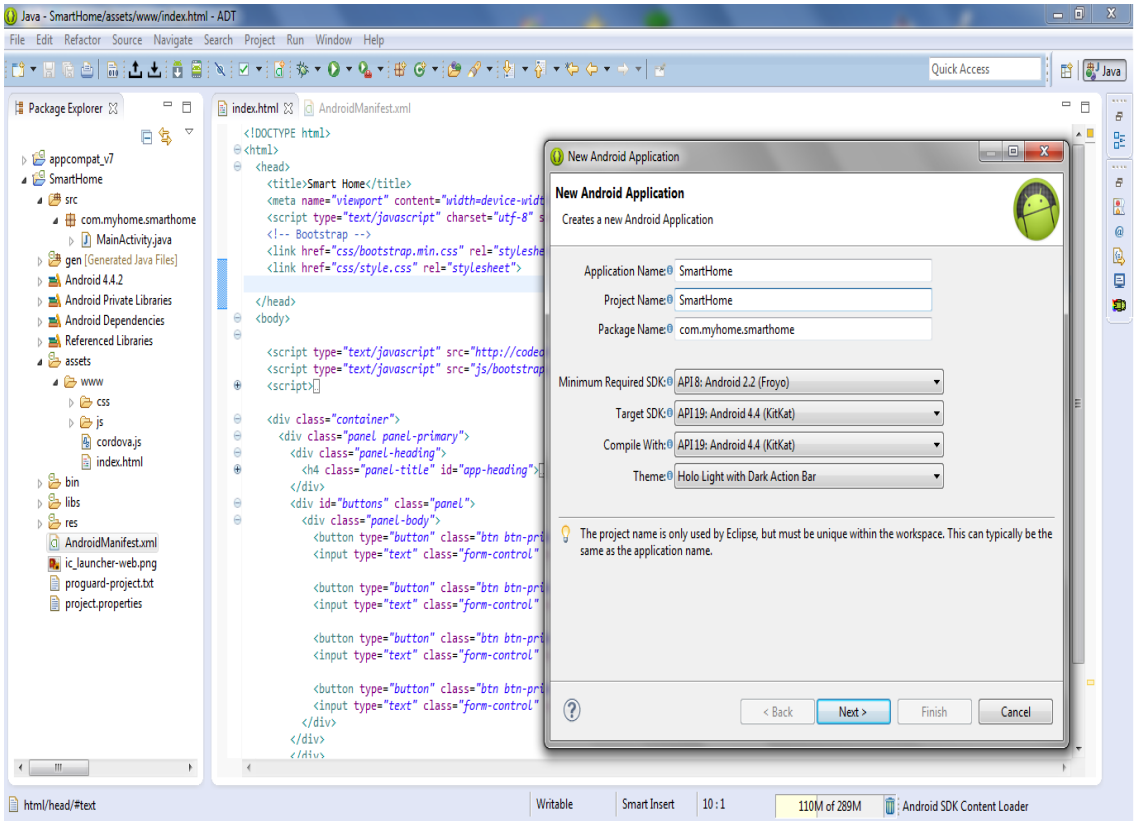
```
        }
      }
//Osnovna razina izvođenja naredbene metode
      function doMethod(method, data) {
         var url = baseURL + coreID + "/" + method;
         $.ajax({
           type: "POST", url: url, data: {
             access_token: accessToken, args: dana
           },
           dataType: "json"
         }).success(function(obj) {
           console.log(obj);
           (obj.return_value && obj.return_value == 200) ?
onMethodSuccess() : onMethodFailure((obj.error)?obj.error:"");
         }).fail(function(obj) {
           onMethodFailure();
         });
      }
      $("#button-1").on("click", function
(){doMethod(funcKey1);});
      $("#button-2").on("click", function
(){doMethod(funcKey2);});
      $("#button-3").on("click", function
(){doMethod(funcKey3);});

//Varijable
      function getVariable(variable, callback) {
         var url = baseURL + coreID + "/" + variable +
"?access_token=" + accessToken;
         $.ajax({
           url: url, dataType: "json"
         }).success(function(obj) {
           console.log(obj);
           (obj.coreInfo.deviceID && obj.coreInfo.deviceID ==
coreID) ? onMethodSuccess() :
onMethodFailure((obj.error)?obj.error:"");
           callback(obj.result);
         }).fail(function(obj) {
           onMethodFailure();
         });
      }
//Dobavljanje metode varijable
      function getVariable1() {
         getVariable(varKey1, function (res) {
           if(res === var1onState)
             $("#var-val-1").val(var1onLabel);
           else if(res === var1offState)
             $("#var-val-1").val(var1offLabel);
           else
             $("#var-val-1").val(res);
         });
      }
      function getVariable2() {
```

```
        getVariable(varKey2, function (res) {
          if(res === var2onState)
            $("#var-val-2").val(var2onLabel);
          else if(res === var1offState)
            $("#var-val-2").val(var2offLabel);
          else
            $("#var-val-2").val(res);
        });
      }
      function getVariable3() {
        getVariable(varKey3, function (res) {
          if(res === var3onState)
            $("#var-val-3").val(var3onLabel);
          else if(res === var1offState)
            $("#var-val-3").val(var3offLabel);
          else
            $("#var-val-3").val(res);
        });
      }
     });
    </script>
     <div class="container">
      <div class="panel panel-primary">
       <div class="panel-heading">
         <h4 class="panel-title" id="app-heading">
           Control
         </h4>
       </div>
       <div id="buttons" class="panel">
        <div class="panel-body">
          <button type="button" class="btn btn-primary btn-lg"
id="button-1">BUTTON 1</button>
          <input type="text" class="form-control" value="---"
readonly id="var-val-1"><br/><br/>

          <button type="button" class="btn btn-primary btn-lg"
id="button-2">BUTTON 2</button>
          <input type="text" class="form-control" value="---"
readonly id="var-val-2"><br/><br/>

          <button type="button" class="btn btn-primary btn-lg"
id="button-3">BUTTON 3</button>
          <input type="text" class="form-control" value="---"
readonly id="var-val-3"><br/>
        </div>
       </div>
      </div>
      <div class="alert fade in" id="info-alert" hidden data-
spy="affix"></div>
     </div>
  </body>
</html>
```

4.11. Izrada mobilne aplikacije za android platformu

Nadalje potrebno je napraviti mobilnu aplikaciju koja će se pokretati na pametnom telefonu. Za izradu mobilne aplikacije koristi se poznato programsko razvojno okruženje "Eclipse" kao što je prikazano na slici 28.

Slika 28. *Eclipse programsko razvojno okruženje*



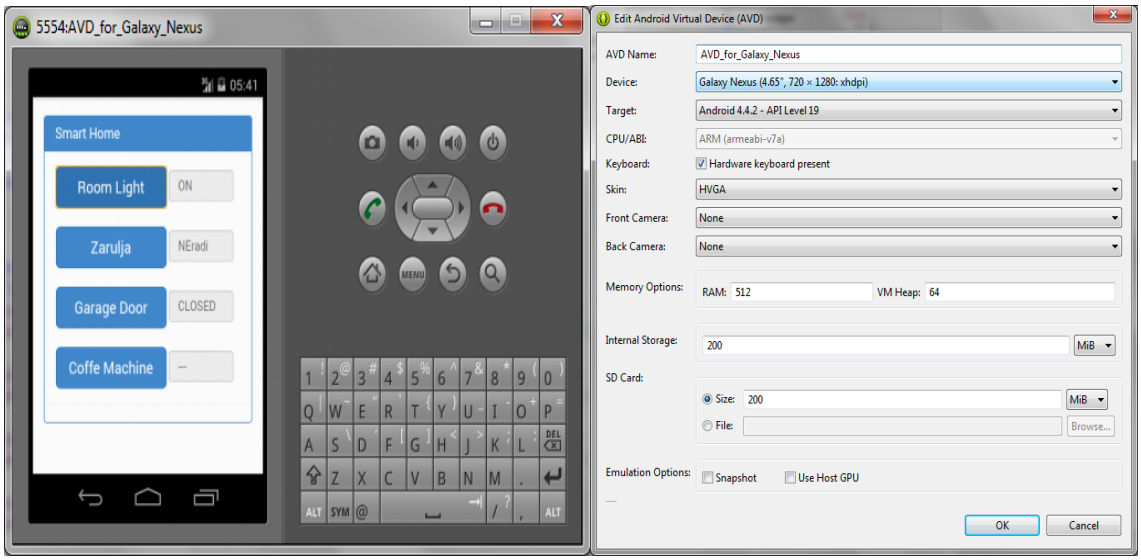
Poradi nedostatka znanja za razvoj mobilnih android aplikacija sa uobičajenim programskim jezicima, u ovome projektu primjenjeni će biti plug-in[XXXIII] u programu "Eclipse" kao što je "PhoneGap"[XXXIV] da bi se mobilna aplikacija mogla bazirati na html programskom jeziku sa dodacima js i css. Na internetu se mogu naći detaljna uputstva kako aktivirati PhoneGap plug-in za njegovo korištenje kao što je "Extending PhoneGap with native plugins for Android" [14].

[XXXIII] Plug-in – U prijevodu priključak, vrsta softverskog dodatka koji ima vlastite knjižnice na raspolaganju, koje nisu dijelom softvera u koji se integriraju. Softver može pomoću tih priključaka i time uporabom vanjskih knjižica pružiti nove funkcije koje nisu bile dijelom jezgre početnog softvera.

[XXXIV] PhoneGap – Besplatan i otvorenog koda framework koji omogućuje stvaranje mobilnih aplikacija putem standardiziranih web APIa (JavaScript, HTML i CSS).

Mobilna aplikacija vizualnog je izgleda kao i web aplikacija sa dodanom funkcijom kao što je Coffee Machine (ISKLJUCENO/UKLJUCENO) koja služi za upravljanje kućnim aparatom za kavu. Nakon izrađene mobilne aplikacije izvršimo njeno testiranje pomoću emulatora, drugim riječima virtualnog android uređaja. Slika 29. koji prikazuje simulaciju pametnog telefona sa željenim preformansama na kojima bi se aplikacija koristila te kako bi izgledala i funkcionirala.

Slika 29. *Emulator sa mobilnom aplikacijom i konfiguracijom*



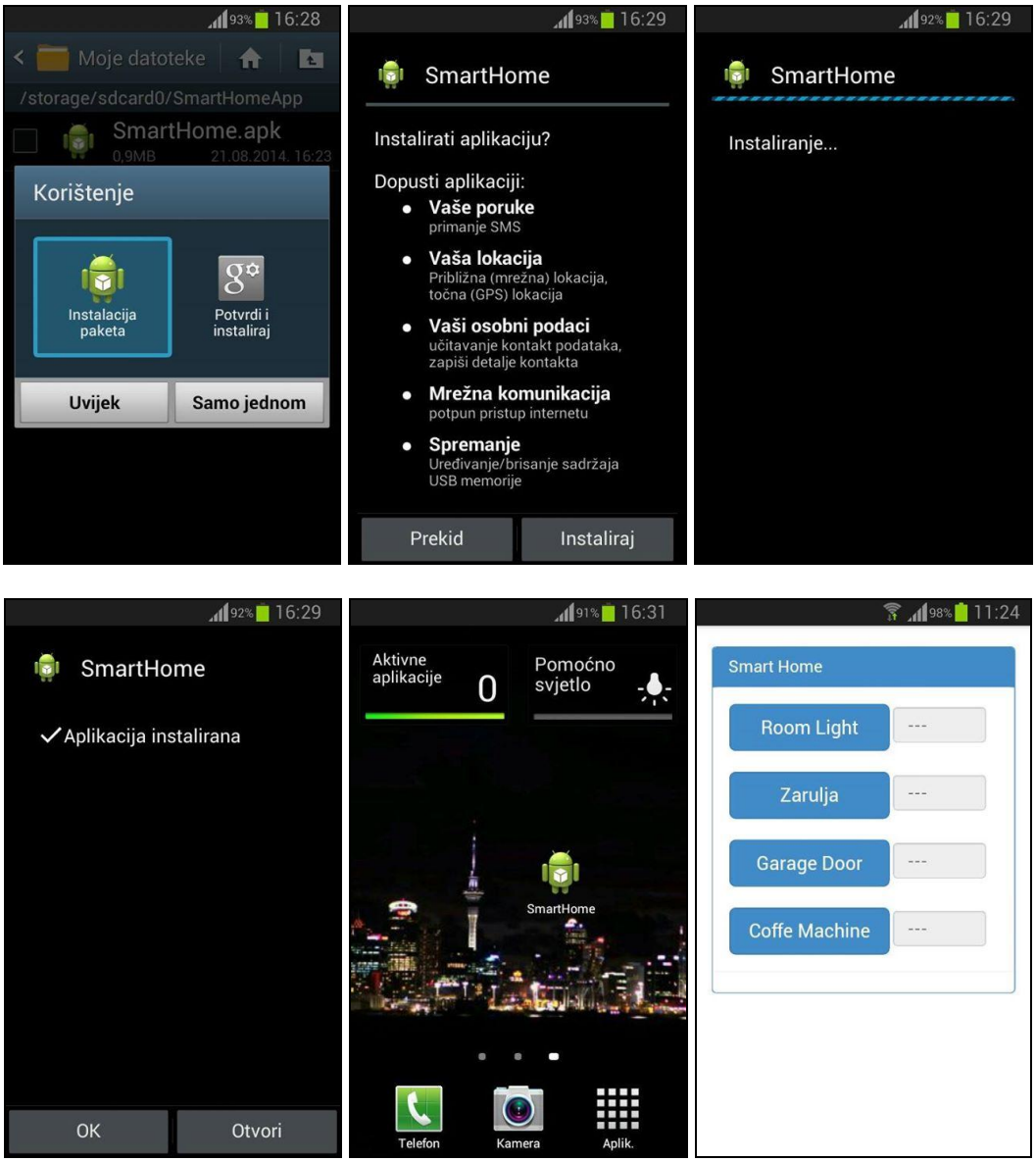
Nakon napravljenog testiranja aplikacije preostalo je da se aplikacija zapakira u format prihvatljiv mobilnom uređaju te se na željeni android uređaj prenese (engl. *Exportira*).

5.TESTIRANJE MOBILNE APLIKACIJE I GOTOVOG HARDVERA

Kada su svi prethodni zadaci u projektu ostvareni, potrebno je izvršiti konačno testiranje cjelokupnog projekta. Kada smo izrađenu aplikaciju pod nazivom "SmartHome" u apk[XXXV] formatu prenjeli na smartphone, potrebna je njezina instalacija kao što je prikazano na slici 30. i pokretanje same aplikacije za rad.

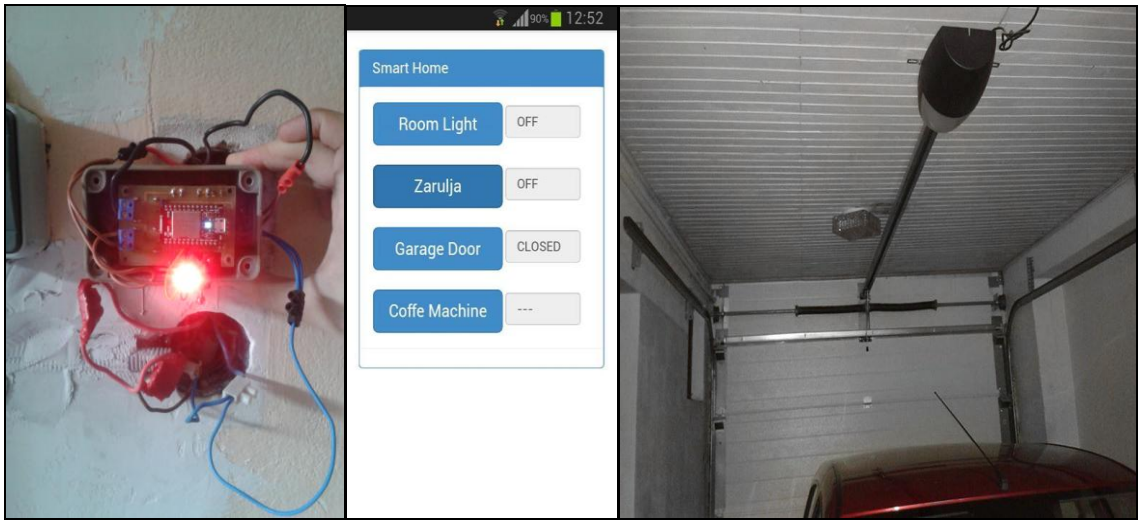
[XXXV] APK – Androidov aplikacijski paket, zapravo paket podatkovnog formata koji se koristi za distribuciju i instalaciju aplikacija za operativne sustave bazirane na androidu.

Slika 30. *Proces instalacije Smart Home aplikacije na pametni telefon*



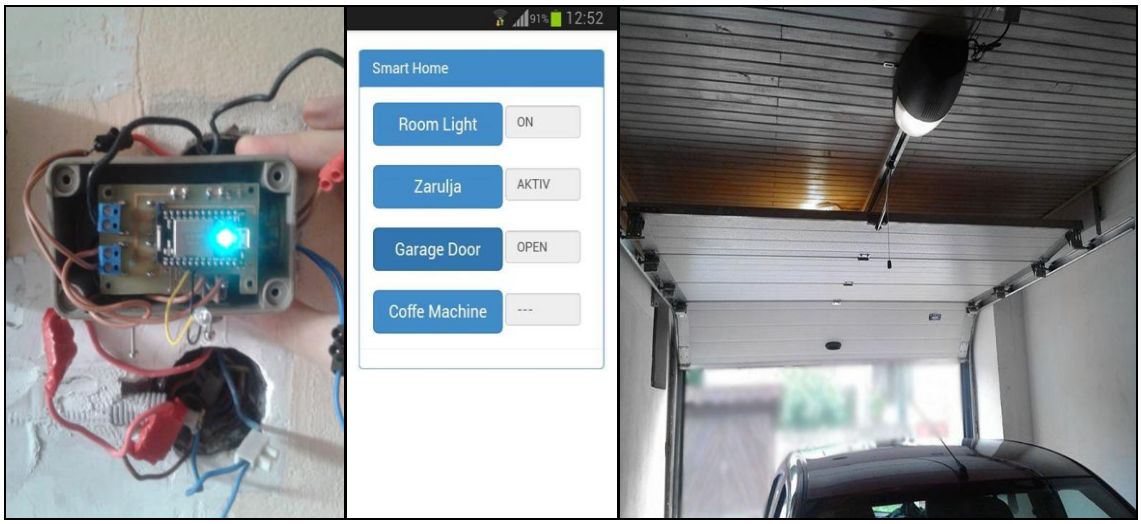
Nadalje je prikazano na slici 31. priključenje hardvera u strujni krug sa priključenom rasvjetom i garažnim vratima, koje je smješteno u jednoj razvodnoj kutijici, te pripojeno na dolazni izvor napajanja od 230 volti. Ujedno su prikazana neaktivna stanja rasvjete i garažnih vratiju, gdje za vizualni prikaz isključene rasvjete (neaktivne žarulje) služi indikator (led dioda).

Slika 31. *Hardver pripojen u strujni krug sa rasvjetom i garažnim vratima u neaktivnom stanju*



Na sljedećoj slici 32. prikazana su aktivna stanja rasvjete i garažnih vratiju (pokrenuta na dodir određene tipke u aplikaciji), te indikator koja ne svijetli ukoliko je žarulja ispravna, kao što je vidljivo sa slike.

Slika 32. *Hardver pripojen u strujni krug sa rasvjetom i garažnim vratima u aktivnom stanju*



Kao što je vidljivo sa prethodne slike kada je svijetlo uključeno (ON) i žarulja svijetli (AKTIV) da će se indikator na razvodnoj kutijici isključiti, ali potrebno je napomenuti, pošto nije vidljivo na slici da ukoliko žarulja u ovome slučaju ne bi svijetlila (OFF), indikator bi bio uključen.

6. PREZENTACIJSKI PROTOTIP UPRAVLJANJA RASVJETOM

Na slici 33. nalazi se DIY projekat sa kućištem što predstavlja razvodnu kutijicu unutar koje se nalaze elektroničke komponente, te sa pripojenim glavnim dovodom napajanja, žaruljom, sklopkom i indikatorom ispravnosti žarulje.

Slika 33. *Prezentacijski DIY projekat*

7. ZAKLJUČAK

Izradom projekta dobilo se praktično iskustvo u programiranju mikrokontrolera, izradi web i mobilne aplikacija uz korištenje programskog razvojnog okruženja prema želji i iskustvo sa elektroničkim komponentama kao i sa strujnim krugovima. Iako je rezultat projekta koristan sustav koji može pojednostavniti svakodnevni život običnog čovjeka, kao što je bilo i spomenuto, da sada više neće biti potrebno ustati iz kreveta ili sa stolice kako bi se na primjer uključilo ili isključilo svijetlo i bilo koji drugi kućni uređaj koji koristi električnu struju za rad. Tako je sa posjedovanjem boljeg znanja moguće i upravljanje sa mnogo složenijim uređajima kao što su TV, radio ili rashladni uređaji pri čemu osim osnovne ON/OFF funkcije moguće je i upravljanje glasnoćom zvuka, tv kanalima, radio stanicama, brzinom i snagom hlađenja i drugim funkcijama. Potrebno je napomenuti da je sam projekt dobar uvod u realizaciju pametne kuće, na način da bi se obuhvatili svi kućanski uređaji kako bi se pretvorili u tzv. pametne uređaje i na taj način, možemo reći da bi cijelu kuću imali u pametnom telefonu, što znači da bi se sa svime upravljalo doslovno sa jednog mobilnog mjesta i to neovisno gdje se nalazili (unutar kuće ili izvan nje, van mjesta, države ili kontinenta) sve dok imamo pristup mreži. Za kraj treba reći kako u ovome projektu postoje neki nedostaci poput manjih dimenzija izrađenog elektroničkog sklopa, kvalitetnije i vizualno bolje mobilne aplikacije, te precizniji sustav detekcije protoka manjih struja do nekoliko stotina mili ampera.

8. LITERATURA

[1] Smurf, L. The Mobi Mag – [How to] Install CM 11 ROM on Samsung Galaxy S3 Mini. http://www.themobimag.com/wp-content/uploads/2013/12/0000229_samsung-galaxy-s3-mini-i8190.jpeg (20.04.2014.)

[2] Blažev, K. mob.hr – Recenzija: Samsung Galaxy S3 mini. http://mob.hr/recenzija-samsung-galaxy-s3-mini (20.04.2014.)

[3] Datatracker.ietf.org – Constrained RESTful Environments (CoRE). http://www.rfc-editor.org/pipermail/rfc-dist/2012-August/003445.html (13.08.2014.)

[4] Waveshare Electronics – STM32F103CB. http://www.wvshare.com/product/STM32F103CB.htm (26.04.2014.)

[5] Texas Instruments – CC3000 SimpleLink™ Wi-Fi Module from Texas Instruments. http://www.ti.com/ds_dgm/images/fbd_swrs126.gif (26.04.2014.)

[6] Texas Instruments – CC3000 SimpleLink™ Wi-Fi Module from Texas Instruments. http://www.ti.com/product/cc3000 (26.04.2014.)

[7] Johnston, S. Wikipedia – File:Cloud computing.svg. http://en.wikipedia.org/wiki/File:Cloud_computing.svg (26.04.2014.)

[8] Masse, M. (2011). REST API Design Rulebook. 1. Izd. Sebastopol, O'Reilly Media naklada.

[9] Spark DOCS – CLAIMING YOURE CORE. http://docs.spark.io/connect/#claiming-your-core (16.05.2014.)

[10] Herrick, R. (2002). DC/AC Circuits & Electronics: Principles & Applications. 1. Izd. Boston, Cengage Learning.

[11] Spark DOCS – THINKERING WITH THINKER. http://docs.spark.io/assets/images/tinker.png http://docs.spark.io/assets/images/tinker-select.png (18.05.2014.)

[12] GitHub – technobly/Remote-Spark. https://github.com/technobly/Remote-Spark (19.06.2014.)

[13] Souders, S. (2007). High Performance Web Sites: Essential Knowledge for Front-End Engineers. 1. Izd. Sebastopol, O'Reilly Media naklada.

[14] Trice, A. Extending PhoneGap with native plugins for Android. http://www.adobe.com/devnet/html5/articles/extending-phonegap-with-native-plugins-for-android.html (11.07.2014.)